Spring 2015

Trajectory generation for lane-change maneuver of autonomous vehicles

Ashesh Goswami
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ashesh Goswami

Entitled
Trajectory Generation for Lane-change Maneuver of Autonomous Vehicles

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

CHUN-SING GEORGE LEE

CHENG-KOK KOH

JAMES V. KROGMEIER

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

CHUN-SING GEORGE LEE

Approved by Major Professor(s): _____

Approved by: Michael R. Melloch 04/28/2015

Head of the Department Graduate Program Date

TRAJECTORY GENERATION FOR LANE-CHANGE MANEUVER OF AUTONOMOUS VEHICLES

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ashesh Goswami

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2015

Purdue University

West Lafayette, Indiana

This thesis is dedicated to my dear parents.

ACKNOWLEDGMENTS

Foremost, I would like to express my gratitude to my advisor Prof. C. S. George Lee, for his guidance and encouragement throughout my research work. I cannot imagine having a better advisor and mentor for my research. I also wish to thank Prof. Cheng-Kok Koh and Prof. James Krogmeier for serving on my Advisory Committee and providing technical guidance.

I am thankful to my lab-mates at the Assistive Robotics Technology Laboratory (ARTLab): Roy Chan, Yan Gu, Andy Park, Debasmit Das and Tongyang Liu for stimulating discussions and their support. I am also thankful to Keerthiraj Nagaraja, my labmate working on a similar research topic for sharing a part of his research work and related software to provide a platform for a portion of my research. I want to also thank my family and friends for their support. I am indebted to them for constant motivation.

Last but not the least, I would like to thank the National Science Foundation (NSF) for sponsoring a part of this research. This work was supported in part by the NSF under Grant CNS-0855098. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF.

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Goswami, Ashesh M.S.E.C.E, Purdue University, May 2015. Trajectory Generation for Lane-change Maneuver of Autonomous Vehicles. Major Professor: C. S. George Lee.

Lane-change maneuver is one of the most thoroughly investigated automatic driving operations that can be used by an autonomous self-driving vehicle as a primitive for performing more complex operations like merging, entering/exiting highways or overtaking another vehicle. This thesis focuses on two coherent problems that are associated with the trajectory generation for lane-change maneuvers of autonomous vehicles in a highway scenario: (i) an effective velocity estimation of neighboring vehicles under different road scenarios involving linear and curvilinear motion of the vehicles, and (ii) trajectory generation based on the estimated velocities of neighboring vehicles for safe operation of self-driving cars during lane-change maneuvers.

We first propose a two-stage, interactive-multiple-model-based estimator to perform multi-target tracking of neighboring vehicles in a lane-changing scenario. The first stage deals with an adaptive window based turn-rate estimation for tracking maneuvering target vehicles using Kalman filter. In the second stage, variable-structure models with updated estimated turn-rate are utilized to perform data association followed by velocity estimation. Based on the estimated velocities of neighboring vehicles, piecewise Bezier-curve-based methods that minimize the safety/collision risk involved and maximize the comfort ride have been developed for the generation of desired trajectory for lane-change maneuvers. The proposed velocity-estimation and trajectory-generation algorithms have been validated experimentally using Pioneer3-DX mobile robots in a simulated lane-change environment as well as validated by computer simulations.

1. INTRODUCTION

1.1 Introduction to autonomous vehicles

Autonomous driving cars have been a long-lasting dream of robotics researchers and enthusiasts. An autonomous car, also known as a driver-less car or self-driving car, is considered as an autonomous vehicle capable of achieving human transportation capabilities of a traditional car. Autonomous vehicles sense their surroundings with on-board sensors and interpret the sensor information to identify appropriate obstacles and navigation paths. Some autonomous vehicles update their maps based on real-time sensor-data, allowing the vehicles to be aware of their positions in unknown environments. The updated map of its surroundings includes obstacles, travel path, and the vehicle's relative position in space. In order to come up with this map, an autonomous car uses sensor equipments such as:

1. Light Detection And Ranging Sensors (Lidar): The continuously spinning lidar system placed on top of the car (costs about $70,000) forms an integral part of the sensing system. The measurements from this sensor are combined with the information from cameras and radar in order to create a detailed 3D map of the surroundings. There also exists 2D laser-scanners to perform simpler tasks of object detection in a single plane of reference.

2. Radar sensors: Radar sensors are mainly used to detect various obstacles (both short range and long range) based on time-of-flight principle.

3. Cameras: Cameras are currently used for lane-detection and obtaining road information. The present day image-processing software can detect traffic signs and lights, lane stripes, cars, etc., and as the technology advances, its importance will increase.

4. GPS Units: Global Positioning System is used for determining a car's location by receiving inputs from satellites.

5. Ultrasound Sensor: Ultrasound sensors are focused on the detection of obstacles around the car during parking.

6. Inertial Measurement Unit (IMU): IMU unit helps with navigation when the signals received from GPS devices are poor.

7. Wheel Sensor: This sensor is used in stability and anti-lock braking systems. It also keeps track of vehicle's location during temporary poor signal reception from GPS devices.

(a) Stanford's Stanley 2005 [1]. (b) Stanford's Junior 2007 [2].

Fig. 1.1.: DARPA autonomous driving challenge participants.

Self-driving cars have marked their presence since the 1970s. In recent years, the Defense Advanced Research Projects Agency (DARPA) has taken a lead on encouraging research in this area. It has organized several competitions for autonomous vehicles (DARPA Grand Challenges) in 2004, 2005, and 2007, respectively. In 2005 autonomous vehicles were able to complete a 131-mile course in the desert. "Stanley", a Stanford-based autonomous car completed the journey in less than 7 hours, emerging as the winner of the competition.

Self-driving cars promise to bring a number of benefits to the society.

1. Safety: Each year, a lot of accidents occur as a result of human errors that can be reduced with increased use of autonomous cars. [3]. Our reaction time is not fast enough to prevent accidents without reducing the number of vehicles. Autonomous vehicles would prevent distractable and sleep-deprived humans from getting behind the wheel.

2. Traffic Management: The preciseness and fast reaction time of machines would help to manage traffic in urban scenarios. By virtue of swarm networks, these cars will also be able to communicate among themselves.

3. Impact on Economy: Autonomous cars will also drive more efficiently than humans, and hence will be fuel-efficient. Insurance costs will go down with reduced number of accidents. The cars would be built with lighter parts, thus reducing construction costs and further improving fuel economy.

4. Better infrastructure layout: Autonomy will make it possible to redesign parking in cities. The ability of a car to park elsewhere after dropping off its passengers would allow parking areas to be moved away from the cities.

5. Mobility to physically challenged drivers: Fully autonomous cars would offer mobility to people who cannot drive themselves, such as the elderly or the disabled.

Manufacturers will use autonomous cars to gather real-world data using different kinds of sensors to compare different algorithms for driving the car. The current driver assistance systems have processors dedicated to sensors for achieving a single task, such as obstacle detection and braking. Cars such as the "Mercedes Benz S-Class" combine several such systems and integrate all the sensory data at a central processor for decision-making. Few years back, the National Highway Traffic Safety Administration (NHTSA) announced standards for car-borne beacons that broadcasts location

information to other vehicles on the road. Information such as position, velocity, and intention to make lane-changes, etc. can be passed on to the neighbouring vehicles.

The lane-change maneuver is one of the most thoroughly investigated automatic driving operations for autonomous vehicles [4]. This maneuver is used as a primitive for performing more complex operations like changing lanes on a highway, merging, or overtaking another vehicle. Most of the accidents during lane-change are caused by failing to leave enough gap, overtaking during poor visibility, or by not giving way to an overtaking vehicle. Fully automated lane-changing operations are realized through different maneuvers such as lane-changing, merging, and overtaking, which have been extensively studied and experimentally demonstrated in the last few years.

1.2 Motivation

Autonomous cars are the future of automobile industry. With the semi-autonomous independent modules already in existence in automotive market, it would not be long before complete autonomous cars become a reality on the road. Google has reported that its self-driving cars have driven more than 500,000 miles without crashing. With such complex systems in the horizon, it is important to pay attention to the intricate details of the entire system. It is necessary for all the smaller independent modules to work properly for smooth functioning of the entire autonomous system.

Autonomous lane-changing is an example of such an independent module of the autonomous vehicle system. There would definitely arise situations for an autonomous vehicle to perform lane-change in scenarios like merging and overtaking on highways. These kind of situations require coordination among the various on-board sensors, path planning algorithms and trajectory-following controllers. To perform a safe lane-change, keeping track of the surrounding vehicles requires a good amount of work to be done in the following domains:

1. Sensing of the environment;

2. Estimation of the position and velocity of the neighbouring vehicles;

3. Trajectory generation for the lane-change maneuver;

Most of the existing algorithms for autonomous lane-changing perform velocity estimation of the surrounding vehicles, plan out a path and execute motion in the planned path. However, they do not take into account sudden changes that might arise in the environment; for example, a change in the velocity of the car being overtaken or may be a different car performing lane-change at the same instant. Thus, an important aspect includes the parameters and factors that need to be taken into consideration while performing the lane-change maneuver. These include distance between the target moving vehicles, velocities of both the target and host vehicles, positioning of the neighbouring cars in their respective lanes, etc. It is important to ensure that even with variations of these factors, the algorithm would have the intelligence to change its decision accordingly and perform a safe lane-change.

With increased amount of research in sensor fusion algorithms on platforms with extensive computation power, achieving a safe lane-change while keeping track of the changes in the surroundings is the main motivation behind this research.

1.3 Related work

1.3.1 Trajectory generation for lane-changing

Lane-change is one of the most common behaviour of vehicles, with applications in numerous road scenarios (Fig 1.2). With recent depth in the research on autonomous vehicles, lane-change has become an active field of research worldwide. The objective of the lane-change maneuver is for a vehicle to move to an adjacent lane under certain constraints.

The traditional algorithms of path planning include graph-search or geometry-based approaches. Dubin curve [6] was a very efficient method to generate a path, but was impractical because of discontinuous path curvature. A different method was based on path-generation using B-spline-curves [7], but it involved complex calculations and resulted in the loss of control of lateral acceleration. Another approach is

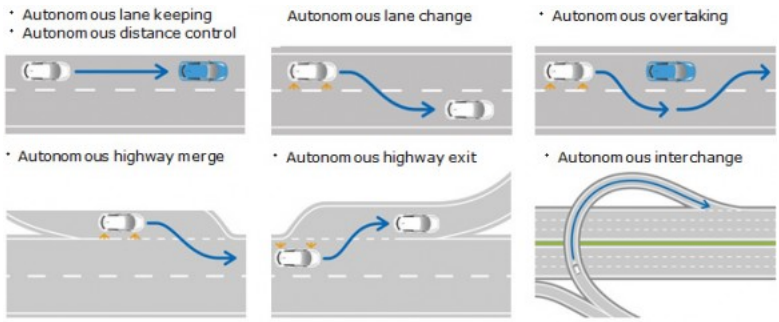
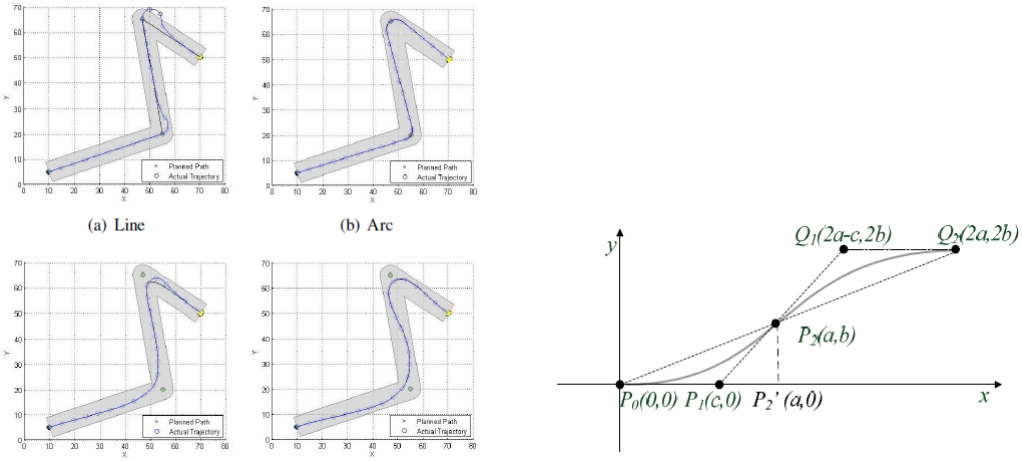


Fig. 1.2.: Different lane-changing maneuvers [5].

based on Bezier curves [8], a fast and efficient path-planning method that is curvature-continuous and satisfies vehicle nonholonomic constraint. Due to the continuity and symmetric property of Bezier curves, piecewise approach of using integration of simpler segments, has become quite popular for path-planning Fig. 1.3.

In [11], a mathematical model and non-linear adaptive controller for a two-vehicle automated overtaking maneuver was developed to consider the problem of overtaking with only relative position information from on-board sensors as feedback. In [12], a two-level architecture was proposed, one for path tracking and the other for lane-change. The high-level layer of the architecture evaluated feasibility of overtaking, and the low-level layer performed appropriate switching from one level to the other.

In [13], study of the automated lane-changing on curved road was performed with unequal non-zero curvatures of the outer and inner lanes. The paper contributes to a trajectory planning method suitable for curved road based on trapezoidal acceleration profile. In [14], the trajectory planning was performed by a two-step algorithm. The first step defined the feasible maneuvers while minimizing the risk of a collision. The second step performed a more detailed evaluation of the feasible trajectories based on additional parameters such as travel time, traffic rules, and comfort.



(a) Bezier curve based path planning [9]. (b) Piecewise Bezier-curve (lane-change [10].

Fig. 1.3.: Application of Bezier curves in path planning.

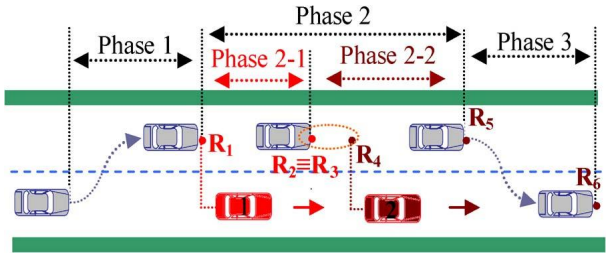


Fig. 1.4.: Stages in an overtaking maneuver [11].

1.3.2 Laser-scanner sensing: (2D and 3D approaches)

Tracking of moving objects from a moving vehicle is a challenging problem and demands a high sensor performance [16]. In recent years, laser range-finders (LIDARs) are being employed to detect and track moving objects. LIDARs are convenient as the motion of the target vehicles can be characterized by position information and

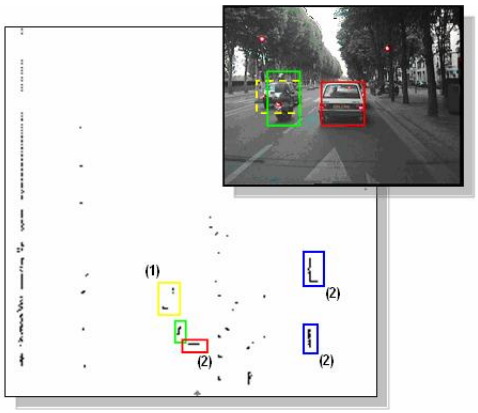


Fig. 1.5.: 2D laser-scan data [15].

its mathematical manipulation [17]. The most popular approaches consist of two independent procedures: detection and tracking. Detection methods need segmentation of data and modeling of objects from extracted features. Usually, tracking methods are based on filtering techniques and an association mtehod, like probabilistic data association (PDA) or multi-hypothesis tracking (MHT). Unfortunately, the object-shape changes during its motion which is sometimes misinterpreted as motion. There exists a necessity to choose time-invariant prominent features to detect target vehicles, otherwise there are possibilities of detecting false apparent motion [18].

In [15], a robust approach for the detection, tracking and classification of multiple vehicles using a vehicle mounted laser-scanner was presented. The classification was based on geometrical configuration with considerations to sensor specifications. The detection and tracking of moving objects is essential for obstacle avoidance and motion planning. For better accuracy, it also becomes important to calibrate and characterize the sensor before its real-time implementation [20].

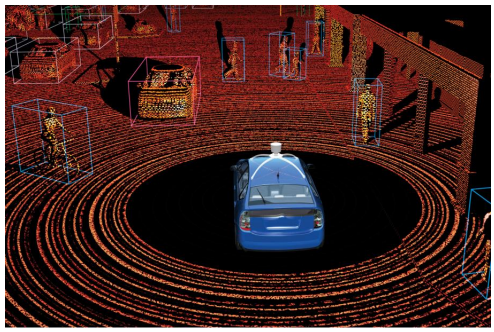


Fig. 1.6.: 3D laser-scanning [19].

1.3.3 Velocity estimation of target vehicles

In [21], a tracking system is set up using 360° field of view LIDARS for detecting the surroundings. The system has the potential to track objects with arbitrary shape (car, pedestrian, bicycle, etc.). The sensor data is segmented into prominent features that are associated with tracked vehicles. A Bayesian filtering is performed to fit a pre-defined vehicle model using sensor data (e.g., extended Kalman filter (EKF) [22], interactive multiple model (IMM) filter [23], and RaoBlackwellized particle filter (RBPF) [24]).

Various attempts at performing tracking of maneuvering targets have been made. The method in [25] assumes known turn-rate, which is quite unrealistic in a highway scenario. Tracking performance deteriorates when the assumed turn-rate differs significantly from the true one. Another approach involves interactive multiple model approach with all possible turn-rates taken into account [26] but the increased number of filters leads to sub-optimal performance. To minimize the number of filters, methods with turn-rate augmented into the state vector [27] have been used for tracking curvilinear motion, sometimes integrating cross-track acceleration as system input [28]. This difficult nonlinear problem requires computationally expensive estimators like

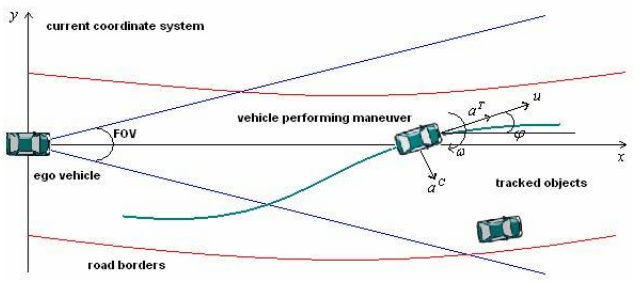


Fig. 1.7.: Velocity estimation of vehicles undergoing complex maneuvers.

Intelligent Kalman Filter (IKF) [29], Unscented Kalman Filter (UKF) [30] or even Particle Filters (PF) [16]. Another approach is to separately estimate the turn-rate (adaptive-grid based turn-rate estimation [31]) and then proceed with the state vector update [32]. In this research, a method of tracking maneuvering target has been proposed that uses a combination of variable and fixed-structure [33] models with an adaptive window-based technique for estimating the turn-rate followed by velocity estimation.

By combining data-association concept with state estimation algorithms, the accuracy of multi-target tracking in noisy scenarios can be improved. Among the various methods, Joint Probabilistic Data Association Filter (JPDAF) algorithm [34] is most commonly implemented in such situations. Different variations of JPDAF [8] have been proposed over the past few years. Instead of common multi-tracking problems of data overlap and coalescence of cluttered data, situations involving continuously changing dataset are faced in a highway vehicle tracking scenario; that is vehicles moving in and out of the sensor-frame. During such scenarios, association of the correct measurement to the appropriate vehicle becomes important. A simplified and revised version of the traditional Nearest Neighbor-JPDAF (NN-JPDAF) method has been proposed for the well-structured lane-classified vehicle data. The association of target measurement pair is done accurately and in a computationally efficient manner.

1.3.4 State-of-the-art technology

Adaptive cruise control

Adaptive cruise control (ACC) for road vehicles automatically adjusts the velocity of the host vehicle to maintain a safe distance from vehicles ahead [35]. It incorporates a simple vehicle estimation followed by locking of the host velocity according to the estimated velocity of the target vehicle. The present-day ACC has a pre-crash system that intervenes to warn the driver during a high risk of collision.

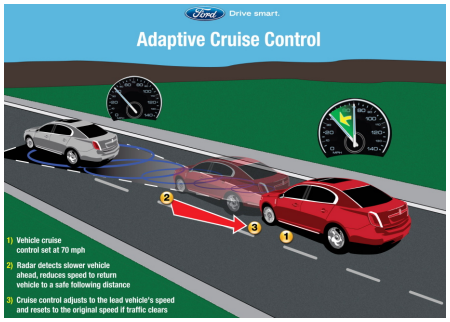


Fig. 1.8.: Adaptive cruise control (installed on a Ford vehicle) [36].

Collision warning & mitigation braking

A collision avoidance system is an automobile-safety system designed to reduce the severity of an accident [37]. The collision mitigating system uses sensors to detect an imminent crash. GPS sensors can detect fixed dangers such as an approaching stop sign through a pre-defined database. In collision mitigation system, sensors detect the possibility of collision and provide a warning to the driver without taking immediate actions. If the vehicle crosses a threshold without any response from the driver, brakes are applied automatically.

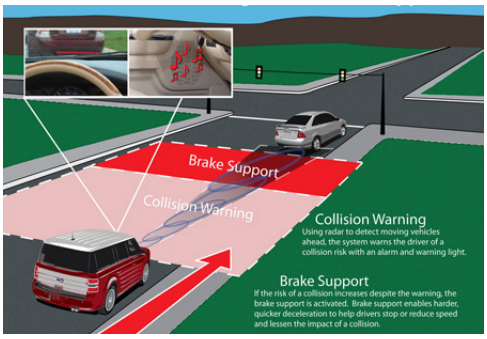


Fig. 1.9.: Collision warning and brake support module [38].

Automated parking

Automatic parking moves a vehicle from a traffic lane into a parking spot, performing parallel, perpendicular or angle parking. The parking maneuver is achieved by a coordinated control of the steering angle and vehicle-speed while performing a collision-free motion into the available space. This is an application of path planning with obstacle negotiation, where all the obstacles are stationary.

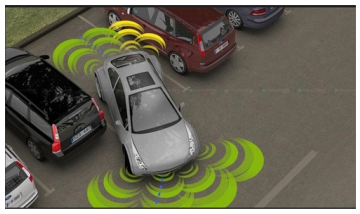


Fig. 1.10.: Automated parking [39].

Lane-keeping & Lane-changing assist

Lane-keeping assist technology alerts the driver when the vehicle deviates from a traffic lane [40]. The vision sensor recognizes the road structure (white lines/yellow lines) and controls the steering to maintain the current lane. It alerts the driver when the vehicle starts to deviate from its lane with a warning buzzer or a small counter-force to the steering wheel.

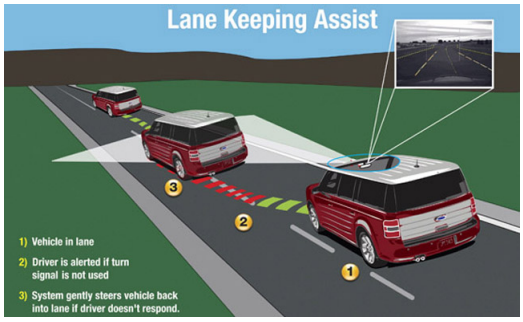


Fig. 1.11.: Lane keeping assistance [41].

In case of lane-changing assist, the system reduces the danger of from the vehicles that approach from the blind-spots. It assists the driver by monitoring of the adjacent lanes [21]. If there is a vehicle in the critical blind-spot area, it is indicated by an illuminating symbol, ensuring attentiveness of the driver.

Google self-driving car

The Google driverless car is a project by Google that involves developing technologies for autonomous cars [44]. The Velodyne 64-beam 3D-LIDAR (worth $70,000) mounted on the top of the car allows the generation of a detailed 3-D map of its environment. The car takes the generated maps, combines them with existing high-resolution maps, thus producing different types of data models that facilitates au-

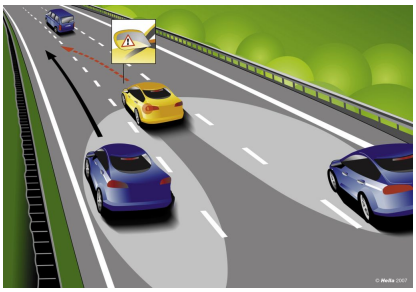


Fig. 1.12.: Lane-changing assist [42].

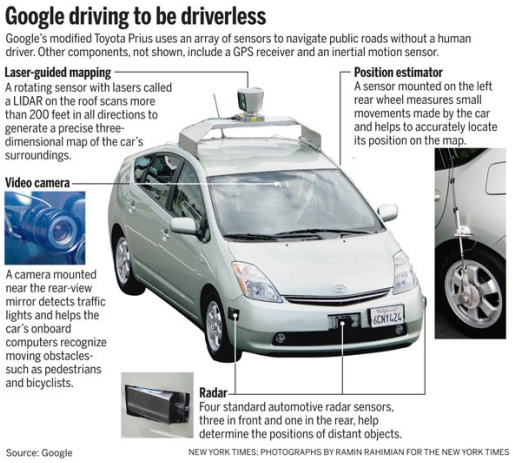


Fig. 1.13.: Google self-driving car [43].

tonomous navigation. Other significant sensors include a camera near the wind-shield to detect road-obstacles, radars for detecting distant objects, position estimators at the rear wheels to gauge the position of the vehicle on a map.

1.4 Research topic, contributions and impact

The research topic involves the task of trajectory generation for lane-changing maneuver of autonomous vehicles. This research attempts to perform trajectory generation based on the performance of three significant sub-tasks; sensing of vehicles in the neighborhood, velocity estimation of the neighborhood vehicles and path-planning of the lane-change maneuver.

One of the problems associated with trajectory generation is the task of vehicle detection in the surroundings. During the motion of the host vehicle, target vehicles need to be detected based on certain characteristic features. In this research, a corner-based feature extraction algorithm has been proposed to perform vehicle detection using a 2D laser-scanner (Chapter 3). The detected vehicles can then be used for tracking, and the tracking information is utilized during trajectory generation. Validation of the algorithm has been performed using experimentation on mobile robot platform using SICK LMS-200 laser-scanner sensor.

Another component involved in the decision-making of a lane-change maneuver is the velocity estimation of the neighborhood vehicles. In this research, a two-stage estimator for multiple target tracking of maneuvering target vehicles has been proposed (Chapter 4). The algorithm provides a tool for overcoming two significant issues; multiple target tracking with appropriate data-association of vehicles moving in and out of the sensor-frame and tracking of maneuvering targets undergoing curvilinear motion. The two-step method eliminates non-linearity involved in state estimations and delivers a high accuracy of turn-rate estimation with very low locking delay between abrupt changes in the turn-rate value. It helps to track the position, velocity and turn-rates of the neighboring vehicles, thus providing information about the sur-

roundings to the host vehicle during lane-change maneuver. The algorithm is not only applicable to lane-changing scenario but can be used for tracking of vehicles at intersections, monitoring over-speeding on highways and many similar applications.

The final component of the algorithm involves the generation of trajectories for performing autonomous lane-chane maneuevers. In this research, a piecewise Bezier curve-based trajectory generation is proposed (Chapter 5). The lane-change maneuver can take place in two modes: one with uniform speed and another with acceleration/deceleration. The trajectories have been generated based on maximization of comfort ride for passengers and minimization of the risks involved during the lane-change. Experimentation with mobile robots in a simulated highway scenario has been carried out to validate the performance of the algorithm.

2. PROBLEM FORMULATION

2.1 Introduction

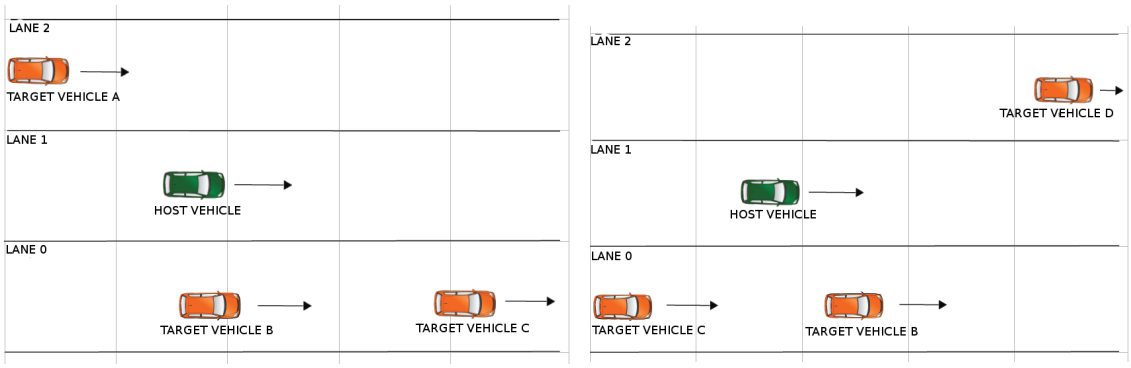
This chapter provides an overview of the different problems that have been dealt with during the research work namely, problem of data-association, turn-rate estimation, trajectory generation and extraction of significant features of the target vehicles. Each of the problems have been explained with illustrative sketches and a brief introduction to the proposed solution has been discussed.

2.2 Problem of data-association

One of the most significant aspects of this research is the problem of data-association during multiple target-tracking of neighboring vehicles. The problem with such tracking is that the vehicles are continuously moving in and out of the sensor-frame and it becomes very important to associate the correct measurement to the appropriate target vehicle. A good data-association algorithm is thus required to keep track of the continuously changing dataset. As shown in Fig. 2.1(a) and 2.1(b), the target vehicles A and C gets replaced by D and E and it becomes necessary to take a note of this change to avoid mis-association. Essentially the problem of data-association can be summarized as the following:

1. To form appropriate target-measurement pairs in a clutter environment of overlapping measurements.

2. To maintain a dataset of tracked vehicles in a dynamic environment (vehicles dropping in and out of the sensor-frame).



(a) Tracking stage 1.

(b) Tracking stage 2.

Fig. 2.1.: Multiple target-tracking problem.

In this research, we have addressed the problem of data-association during autonomous navigation. The main objective is to reduce time and computation complexity of the problem while simplifying the target-tracking algorithm by classifying the measurement into a well-structured dataset.

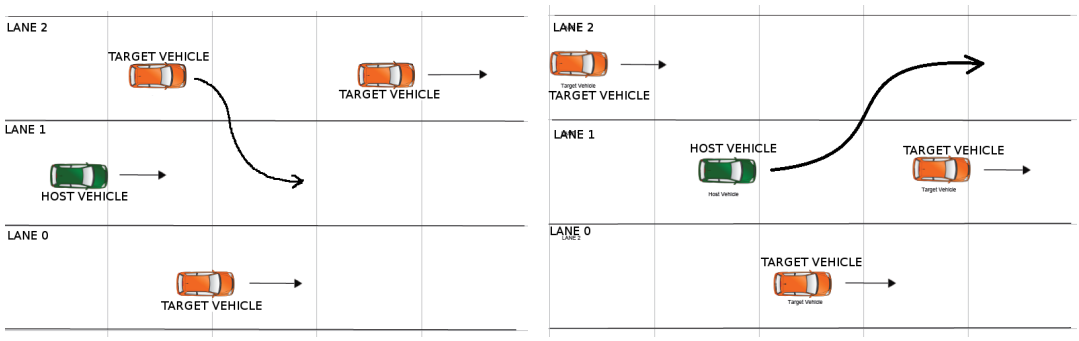
1. A combination of deterministic (Nearest Neighbor Standard Filter) approach and probabilistic approach (Joint Probability data-association Filter) has been undertaken to reduce the mean squared error of the measurements.

2. Linked-list based data structure has been maintained for ease of manipulative operations like insertion and deletion of target vehicles.

2.3 Problem of turn-rate estimation

Maneuever target-tracking is essential during situations where either the host vehicle or the target vehicle, or both are undergoing lane-change. During the curvilinear velocity estimation, non-linearity is introduced. The proposed algorithm provides an

alternative to performing the non-linear computations, thus reducing the complexity of the tracking algorithm.



(a) Host undergoing lane-change (b) Target undergoing lane-change

Fig. 2.2.: Curvilinear velocity estimation scenarios.

As shown in Fig. 2.2(a) and Fig. 2.2(b), velocity tracking needs to be performed even while vehicles are performing lane-change. The problem of turn-rate estimation can be summarized as:

1. Estimation of the turn-rate of the target vehicles while tracking their positions and velocities with high accuracy.

2. Avoiding non-linear computation during velocity estimation of target vehicles.

3. Detection of abrupt changes in turn-rates within a short period of time.

A solution to the above problem has been proposed using a maneuvering target-tracking algorithm based on the lines of Adaptive Grid (AG) method [45]. Fixed and variable values of turn-rates have been utilized in dividing the turn-rate space into several windows, and then converging towards the true turn-rate value. The solution has been validated for situations involving tracking of highly maneuvering targets using stationary sensors as well as vehicle tracking by a moving host vehicle.

2.4 Problem of lane-change trajectory generation

An important aspect of this research is the trajectory generation during lane-change. The lane-change scenario can be a simple one with just one vehicle in the same lane (Fig. 2.3) or a complex one with multiple vehicles in the neighborhood (Fig. 2.2(b)). The algorithm considers the relative position and velocity of the neighboring vehicles and decides about the feasibility of a safe lane-change. Based on the decision, appropriate trajectories are generated. It is to be noted that there is a difference between a lane-change maneuver and an overtaking maneuver. An overtaking maneuver is a sequence of a lane-change maneuver, a path tracking along the new lane, and a return to the original lane, thus requiring a higher degree of planning and coordination ([12], [46])

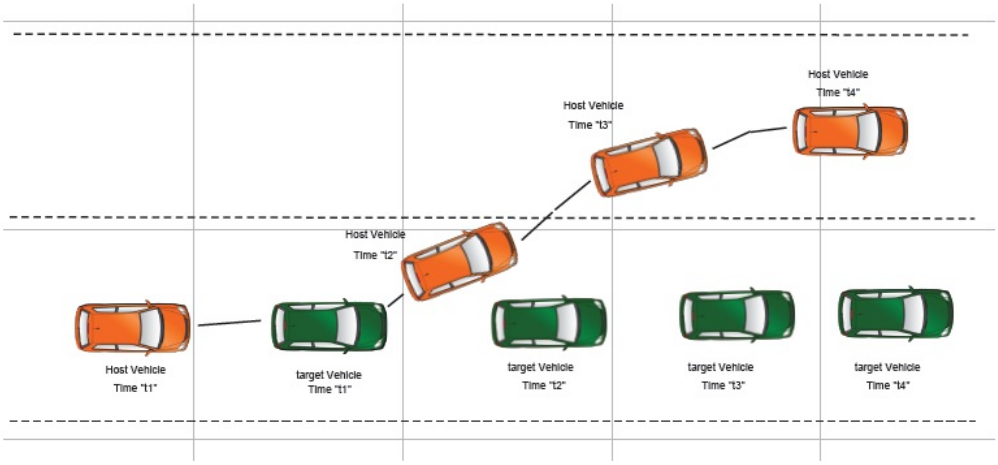


Fig. 2.3.: Simple lane-changing scenario.

Summarizing the different aspects of these situations, we can formulate the problem as:

1. Decision-making regarding the feasibility of lane-change based on the information about the surroundings.

2. Generation of a trajectory based on decision, while minimizing the risk involved and maximizing the comfort-ride parameter.

The proposed algorithm has options for both advisory and co-pilot modes providing either a set of feasible paths or the most optimal one. The generation of trajectory is done in two modes: uniform-speed and accelerated mode based on the situation at hand (discussed in Chapter 5).

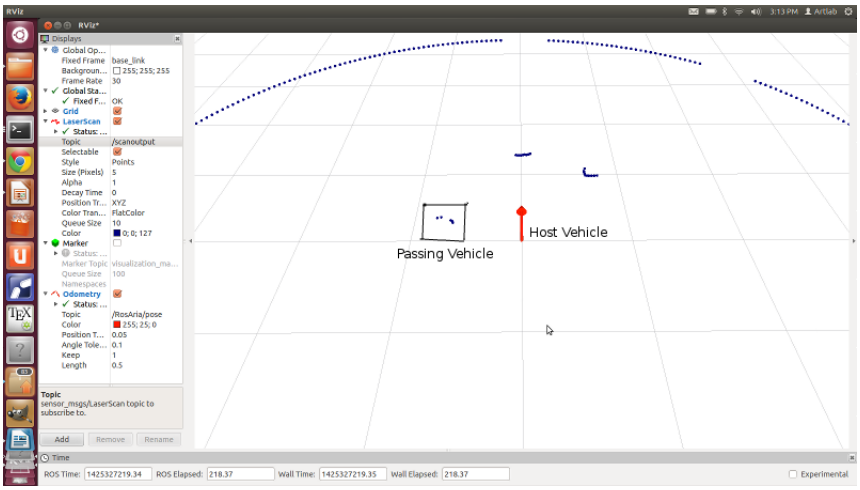
2.5 Problem of feature extraction

A part of the research focusses on the detection of vehicles in the given scenario. It involves extraction of significant features of the target vehicles that can be used for tracking purposes. The challenge with such extraction algorithms is that unless a prominent feature is taken into consideration, tracking the vehicle becomes difficult. Figure 2.4 demonstrates an example of the change in appearance of a vehicle as it passes by the host vehicle. A reliable shape-invariant feature is thus required for correspondence matching. A summary of the vehicle-detection phase of the research is as follows:

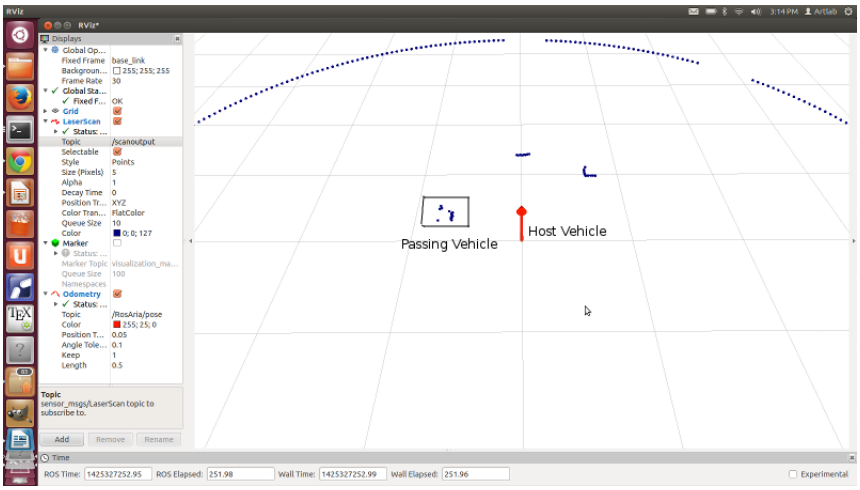
1. Detection of significant features of the target vehicles for tracking purposes.

2. Obtaining a well-structured classification of the vehicles for simplification of the data-association problem.

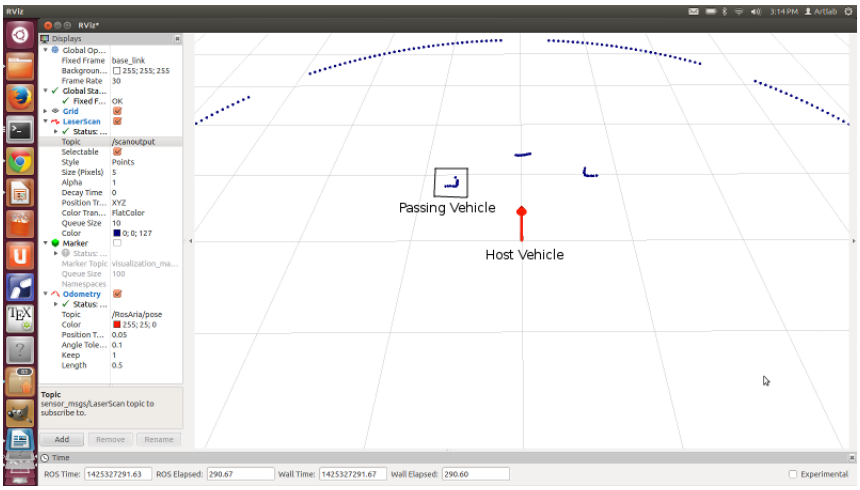
Some of the commonly used algorithms have been implemented for segmentation of road-data and detection of lanes. Corners of the vehicles, if visible, have been used to identify the vehicles over time using a modified Douglas Pecker algorithm (Section 3). Classification of the measurement data into corresponding lanes has been achieved by implementing camera-based lane-detection algorithm.



(a) Shape of vehicle at time t_1.



(b) Shape of vehicle at time t_2.



(c) Shape of vehicle at time t_3

Fig. 2.4.: Appearance of a passing target vehicle.

2.6 Important assumptions

With the complex nature of the lane-changing problem, it becomes necessary to make some assumptions about the problem. The assumptions are mostly concerned with the lanes, traffic scenarios and some of the vehicle parameters taken into consideration.

1. **Lane Characteristics:** The roads are assumed to be straight; that is, the road curvature is minimal and hence any curvilinear motion observed for a vehicle is due to its trajectory and not the curvature of the road. Standard lane dimensions of 3.7 meters (9.8 feet) width has been considered.

2. **Vehicle Parameters:** The vehicle speed is assumed to be within a range of 5 m/s - 25 m/s (11.2 mph - 56 mph) which is quite a reasonable assumption in highway scenario. A maximum acceleration of 1.5 m/s^2 (4.9 $feet/s^2$) and maximum of 20° turning angle during lane-change have been assumed.

3. **Scenarios:** It has been assumed that there is no bidirectional traffic. Also, highway scenarios have been considered during the validation of the algorithm which reduces the number of non-vehicle objects, and thus any detected object on the road can be considered as a target vehicle.

2.7 Functional block diagram of the proposed lane-change algorithm.

Based on the problems faced during this research, an overall block diagram of the proposed lane-changing algorithm is shown in Fig. 2.5. As seen in Fig 2.5, the algorithm comprises of three components, namely, vehicle-detection, target-tracking and trajectory generation. The sensor-data is subjected to certain filtering algorithms to eliminate noisy data. Vehicle-detection is then performed to extract features from the vehicles and classify them according to their respective lanes. The next step is the data-association algorithm where the measurement data is paired up to their respective target vehicles. Once the individual pairings are completed, state-vector

update is performed using the proposed velocity estimation algorithms. Data from the tracking algorithm along with other constraints are utilized for generating lane-changing trajectory (Section 5).

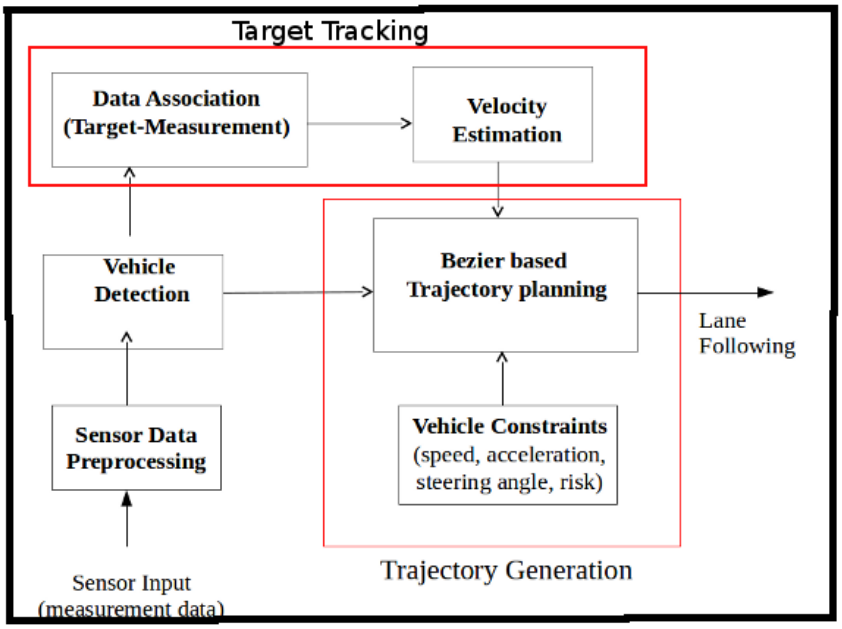


Fig. 2.5.: Functional block diagram.

2.8 Development details

As discussed previously in this section, this research involves the development and analysis of the proposed autonomous lane-change algorithms. The following gives a brief overview of the development tools that have been used during the research to validate the propsed algorithms:

2.8.1 Software development using Robot Operating System (ROS)

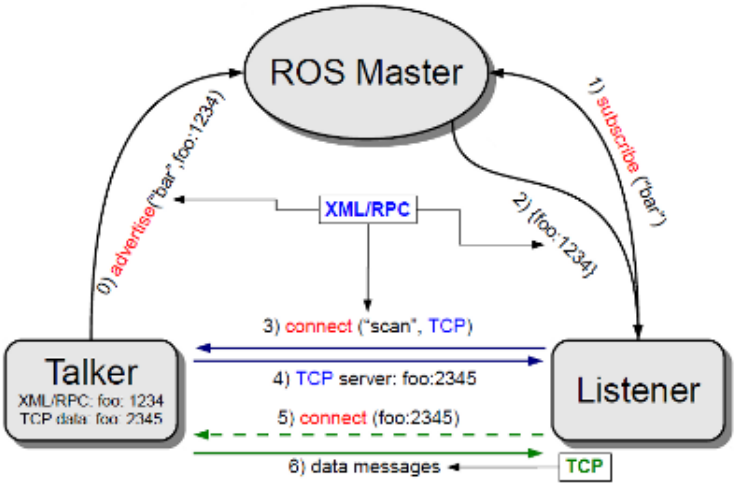
Introduction to ROS

In order to be modular and ensure independent functioning of each package, ROS follows a multi-process architecture. A process performing a specific action is called node. Such nodes communicate with each other by passing messages, where a message is a data structure composed of primitive data types like integers, strings, arrays etc. A topic to which nodes publish messages serve as a buffer to store the communicated messages. Multiple publishing nodes can write to a topic and multiple subscriber nodes can subscribe to a topic, a typical example of a broadcast type communication. The topic is implemented with a first-in-first-out (FIFO) protocol (depicted in Fig 2.6). ROS is designed in lines of distributed computing and can be used across multiple machines. Transmission Control Protocol (TCP) is used for ensuring reliable communication across multiple processes. TCP is optimized for accurate delivery but not for timed delivery, hence it can cause delays [47].

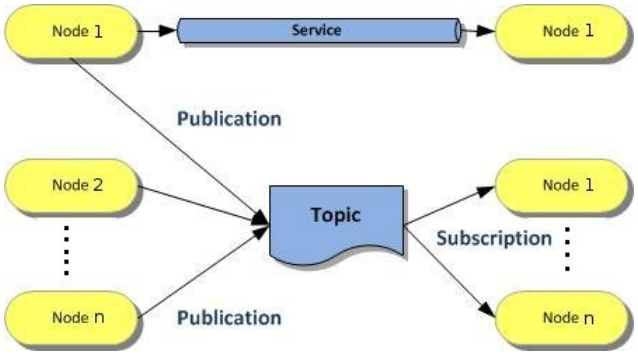
In spite of ROS exhibiting non real-time behavior, it is widely used for controlling robots that do not require real-time response. Assembly tasks in warehouses, execution of motion planning algorithms, navigation in unknown environments are some common non real-time robotic applications. ROS is an open-source environment with increasing user-community and hence has a lot of available support. Due to these advantages, it is convenient to validate the results of proposed algorithms using ROS, especially when the goal is not real-time implementation.

Application of ROS for validation of developed algorithm

For experimentation purposes, software development of both the target-tracking and lane-changing algorithm has been carried out using ROS. The following are some of the reasons why ROS has been chosen as a platform for the validation of the developed algorithms:



(a) Standard talker-listener communication [48].



(b) FIFO based publication-subscription

Fig. 2.6.: Overview of ROS nodes communication [47].

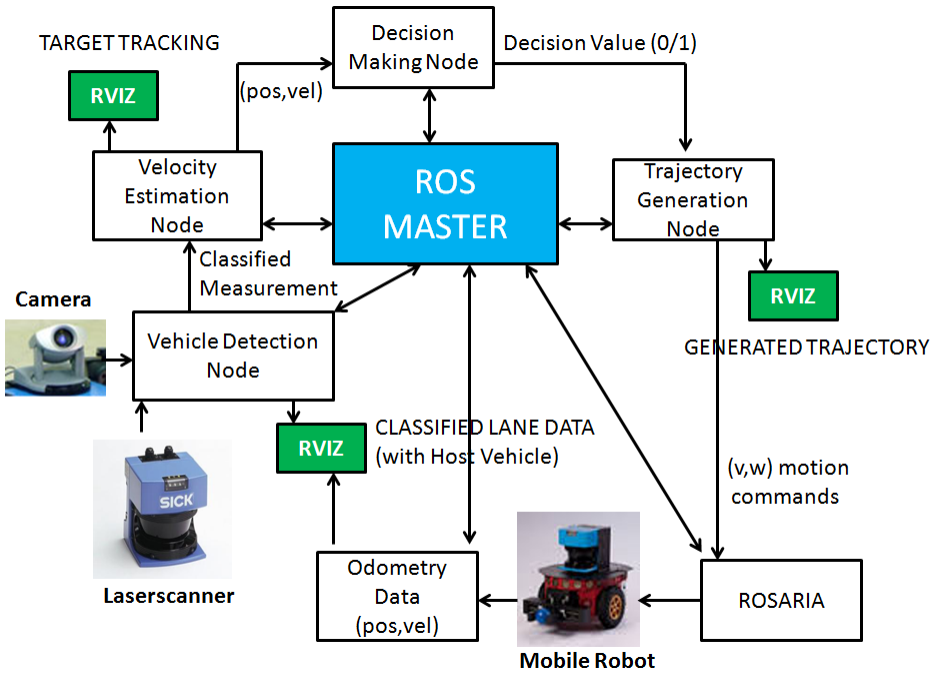


Fig. 2.7.: Communication among ROS nodes during experimentation.

1. Separate nodes have been created for different aspects of the algorithm that can communicate with each other as well as ROS-master using messages.

2. Dependency of information among the nodes is demonstrated, especially during the decision-making regarding feasibility of lane-change.

3. Simulation using Rviz tool for lane-classified data, target-tracking as well as path-planning has been performed.

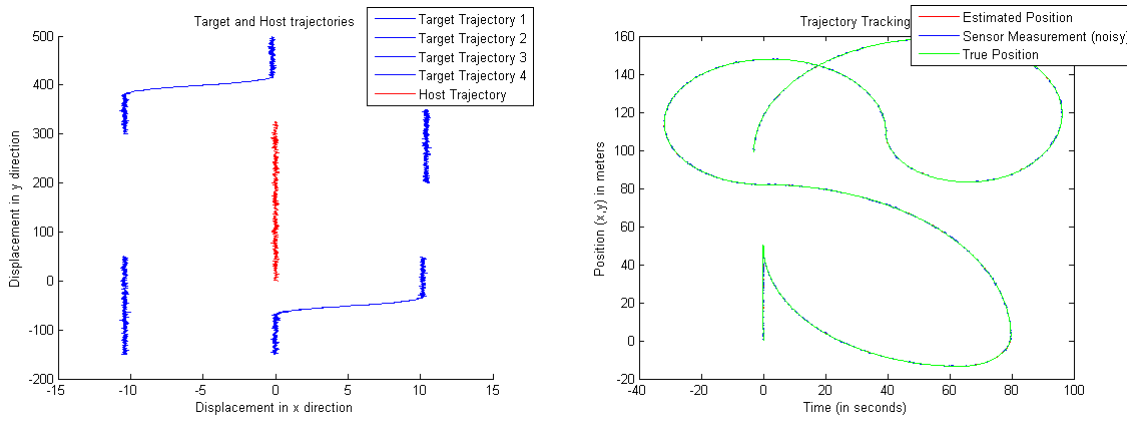
4. P3dx (robot) motion control is performed using ROSARIA, a service dedicated to provide motion commands to the robot.

Parallelization of tasks, flexibility of communication between the nodes, availability of drivers for sensor communication and a good visualization tool make ROS a good platform for validation of the developed algorithm. However for a real-time

implementation, a Real Time Operating System (RTOS) based ROS [49] has to be utilized.

2.8.2 Simulation using MATLAB

MATLAB has been extensively used for simulation purpose of the proposed target-tracking algorithm. Various test scenarios have been simulated to validate the data-association and curvilinear velocity estimation aspects of the algorithm. Numerous plots demonstrating several aspects of the estimator outcomes (errors, position and velocity tracking, turn-rate estimation, etc) have been illustrated. The advantage of MATLAB simulations is that most of the mathematical operations are dealt with much ease before interaction with the embedded controllers. Figure 2.8 demonstrates a couple of such scenarios that have been utilized for the validation of the proposed algorithms. Further details regarding the same have been provided in Section 4.



(a) lane-changing scenario.

(b) Maneuvering target-tracking using stationary sensor.

Fig. 2.8.: MATLAB target-tracking simulation scenarios.

2.8.3 Hardware interface

1. **Pioneer3-DX mobile robots**: For validation purposes, an artificial lane scenario has been simulated in the laboratory using P3-Dx mobile robots as our autonomous vehicles. Figure 2.9 shows the experimental set-up in Assistive Robotics Laboratory for performing target-tracking and autonomous lane-changing. Fig 2.10(a) shows the mobile robot being used for this purpose. ROSARia is the platform that has been used for interfacing the software with the robot using RS422 communication protocol. Speed and performance of the algorithm can be improced by upgrading the hardware components used during the experimentation.

Fig. 2.9.: Simulated lane-change evironment in Assistive Robotics Laboratory.

2. **2D LMS Sick laserscanner:** SICK 2D laserscanner has been used as the sensor for measuring the position of the target vehicles. It is installed on the robot and covers a field of view of 180 °about its center and a linear range of

(a) Pioneer3-DX mobile robot [50]. (b) SICKLMS200 scanner [51].

Fig. 2.10.: Pioneer3-DX mobile robot with laser-scanner.

about 50 meters. All the vehicles are assumed to be at a height such that it is in the field of view of the sensor plane.

2.9 Summary

In this section, a detailed description of the different problem scenarios faced during the research has been provided. As mentioned, the lane-changing scenario involves three significant areas of research: scanning of the environment, tracking of the neighborhood vehicles and generation of the lane-changing trajectory. The requirement of further research into each of the categories have been well justified. An approach to obtain the solutions for each of the problems have been briefly discussed. An overview of the developed software architecture has been illustrated (ROS). An introduction to the simulation tools and experimentation platform has been provided to familiarize the reader regarding the validation methods used during the research. The following sections discuss in detail each of the components of lane-changing algorithm and provide the validation results based on experimentation outcomes.

3. SENSING AND VEHICLE DETECTION

3.1 Introduction

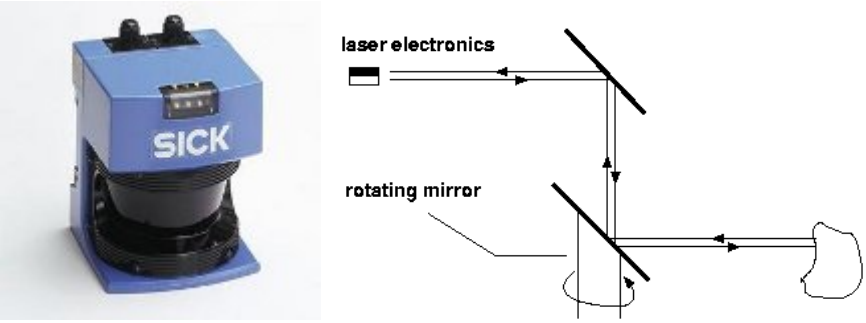
In this chapter, the significant role played by a laser-scanner during vehicle detection will be discussed. A detailed overview of the 2D laser-scanning process along with modeling of sensor characteristics will be provided. Some of the common problems faced during scanning will be discussed, followed by proposal of methods to overcome them. A corner-detection based feature-extraction algorithm has been proposed to perform vehicle detection during target-tracking. The algorithm is a variant of the existing method of polyline simplification using Douglas-Pecker algorithm. Vehicle detection forms the first stage of tracking target vehicles in the neighborhood of the host vehicle (discussed in Section 4). Validation of the algorithm using experimentation on mobile robot platform has been carried out and discussions based on the results will be provided in this chapter.

3.2 Vehicle detection using 2D laser-scanners

3.2.1 Functionality of 2D laser-scanners

A laser-scanner is an active optical position measurement sensor. Using the popular time-of-flight measurement principle, a laser pulse is sent out by the sensor that reflects off of an object. The elapsed time before the pulse returns is converted into distance. In a scanning laser-scanner, motion of a scanning mirror directing sequential pulses in different directions creates an approximated 2D model of the environment.

The laser-scanner gives the range ρ and azimuth angle θ for every obstacle point in its field of view. The scanning mirror moves continuously, but as the measurements are taken at discrete angle increments, azimuth error is introduced. For a given beam



(a) SICK LMS-200 scanner [51].

(b) Time-of-Flight (rotating mirror) concept [52].

Fig. 3.1.: 2D laser range-scanner .

angle on the target, the position uncertainty is proportional to the range, the error in which results from the per-pulse range measurement process.

The system under consideration for the lane-changing application uses SICK LMS-200, a single-scan axis scanning laser-scanner. The scanner is oriented to scan in a horizontal plane (using 2D geometry). Performance specifications are 1 cm range resolution, 50 meter range, 1° basic azimuth resolution and 75 Hz frequency. The angular resolution of the scanner is selectable at 1 inch, 0.5 inch, or 0.25 inch. However, the 0.5 inch and 0.25 inch resolutions are achieved only by interlaced scanning; that is, for each individual scan, the physical resolution is 1 inch. Although higher angular resolution is desirable, time-skew between adjacent points leads to inaccuracies at higher velocities. Hence, we limit our interest to the 1 inch resolution setting, where a full scan of 180° produces 181 measured range values. The data transfer rate can be programmed to be 9.6, 19.2, 38.4, or 500 kilo-baud according to the hardware constraints.

3.2.2 Role of vehicle detection in target-tracking

The goal of vehicle detection is to identify moving objects and determine their positions using a laser-scanner sensor. A track is the identity of an object coupled with estimates of motion parameters over change of time. The tracker generates these tracks from a series of measurements at a certain frequency.

1. **Object correspondence**: We need to establish feature-based object correspondences between consecutive measurements to estimate motion parameters (velocity and acceleration).

2. **Object identity**: The object identity is useful to detect objects when they appear into and disappear from the sensor-frame.

Vehicle detection thus plays a significant role in segmenting out the vehicles and performing feature correspondence to associate the correct measurement to the appropriate target vehicle [18].

3.2.3 Common problems of tracking using laser-scanner

As with any other vision sensors, laser-scanner also has a number of problems associated with it that arise during vehicle-tracking. The following six problems are commonly faced during tracking of the vehicles in motion:

1. **Motion of sensor**

   A crucial aspect of the detection problem is that the laser-scanner is in motion, and hence the data requires continuous coordinate transformation. Even after the position is corrected by a coordinate transformation, its appearance changes during motion due to the changing scanner perspective. The shape-change makes it difficult in determining whether these changes are due to changes in perspective or from actual motion of the tracked vehicle.

2. **2D scan of a 3D world**

   Three major problems arise from using a single axis 2D laser-scanner:

(a) **Contour**: During instances of pitch/roll of the scanner, different portions of the vehicles are visible.

(b) **Ground reflections**: When the ground is not flat, the scanner beam may hit the ground and consider it to be an obstacle. Furthermore, due to pitch/roll, these returns may appear to be be in motion.

(c) **Obstacle height**: Based on the sensor's position, we can detect obstacles that are taller than the height of the sensor from the ground. If the target vehicle is at a level lower than this height, it may be completely missed during the scan.

3. **Segmentation**

   As the laser-scanner provides information at discrete points on a 2D plane of the vehicle surface, segmentation of vehicles from the overall scene based on this scarce information is very challenging.

4. **Clutter**

   Objects being close to each other induces the need to associate the correct target with its corresponding measurement. It can also cause spurious disappearance of valid targets, for example a pedestrian moving close to a wall may appear to be merged with the wall.

5. **Weak returns**

   Some objects have very poor reflectivity at the operating frequency of the scanner and can result in potential loss of data.

6. **Mixel-pixel problem**

   When a laser-spot is located at the edge of an object, the measured range is a combination of the foreground object and the background object, a condition commonly referred to as the mixed-pixel problem.

3.2.4 Sensor modeling

In this section, an experiment aimed at characterizing the laser-scanner has been performed. We have analysed the effect of two operating parameters (range and incidence angle) on the sensor measurements.

Range-measurement model

The range-measurement model is a simulation model that estimates the true distance using the measured range [20]. It may be used to simulate the functionality of a scanner during simulations. There are two basic approaches towards this problem:

1. Linear approximation: We try to approximate a linear function that maps the measured distance into the true range. This involves propagating the measured range through a linear transformation

$$\hat{\eta} = n_1 r + n_2 \ , \tag{3.1}$$

where $\hat{\eta}$ is the estimated mean range, r the measured range and n_1 and n_2 being constants defined as:

$$n_1 = \frac{\sum_{i=1}^{n}(r_i - \bar{r})(\eta_i - \bar{\eta})}{\sum_{i=1}^{n}(r_i - \bar{r})^2} \ , \tag{3.2}$$

$$n_2 = \bar{\eta} - n\bar{r} \ . \tag{3.3}$$

where r_i and η_i denote i^{th} measurement range and mean range respectively, $\bar{r}$ and $\bar{\eta}$ representing the mean values of r and η respectively. Assuming the measurement error is Gaussian noise, the measured range value of a single scan is $Z = int(n_1 r + n_2 + \gamma)$, where γ is Gaussian noise with a normal distribution $N(0, \sigma)$ with σ representing the variance, the rounding function int(.) denotes the integral value of the computed sum.

2. Computing the noise covariance matrix R: This involves collecting sensor-data with stationary host and target vehicle over a period of time and computing the joint probability distribution of the data in Cartesian form using

$$x = r\sin\theta \ ; \qquad y = r\cos\theta \ , \tag{3.4}$$

where r and θ denote the range and angle measurements respectively.

Once the joint probability distribution is made, marginal distribution of the two directional components of the sensor-data are obtained separately and used to compute the individual expectation and covariance. Using the values obtained, we compute

$$R = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \ , \end{bmatrix} \tag{3.5}$$

where R represents the noise covariance matrix with σ_{xx} and σ_{yy} being the auto-correlation terms and σ_{xy} and σ_{yx} being the cross-correlation terms.

Effect of incidence angle

In this section, we examine the effect of incident angle on the range measurement accuracy. In the experimental set-up, the host vehicle is used to track several stationary target vehicles located in its proximity. The errors appear to be minimum when objects appear right in front of the host vehicle ($60^\circ - 120^\circ$). It increases towards the edges of the objects due to the mixed-pixel problem when the laser strikes at an angle to the obstacle. Errors appear to be maximum somewhere near $25^\circ - 35^\circ$ as well as $170^\circ - 180^\circ$. Figure 3.2 depics the variation of errors over the entire 180° range of measurements.

3.3 Vehicle detection algorithm

The process of vehicle detection comprises of the following stages:

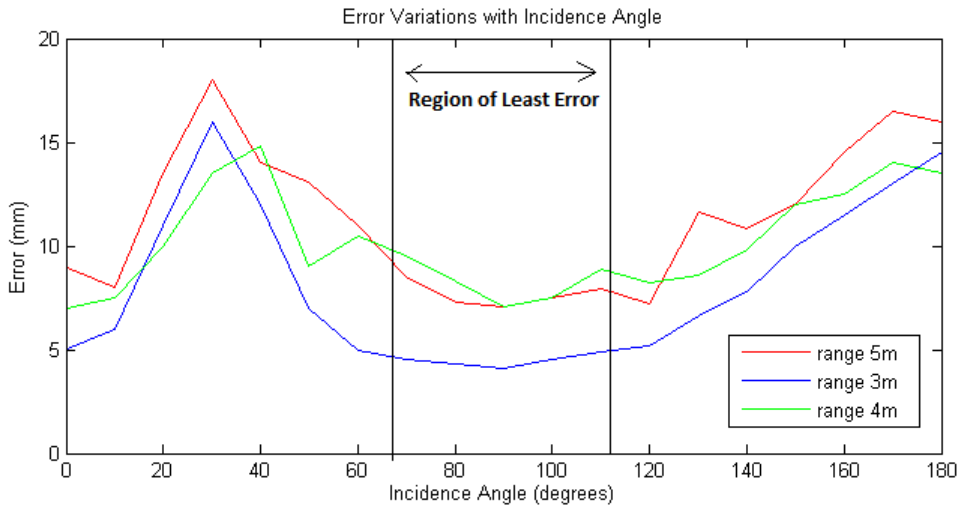


Fig. 3.2.: Error variations with incidence angle of laser beam.

1. Clustering of detected objects: It involves segmenting out the objects on the road by clustering the scan-points followed by assignment of basic shapes to each of the clusters.

2. Extraction of data within the lanes: It involves segmenting out the road information by removing vegetation, cross-walks and other data irrelevant to the target-tracking algorithm.

3. Classification of vehicles in respective lanes: It involves assigning lane labels to each of the vehicles based on their respective positions.

4. Feature-extraction: It involves corner detection of each of the vehicles, performed in two stages as follows:

   (a) Horizontal and vertical clustering of scan-points: It involves a variant of Douglas Pecker's algorithm (Section 3.2) to cluster the scan-points of each vehicle into the visible longitudinal and lateral clusters.

(b) Corner detection: It involves line fitting through the points in the longitudinal and lateral clusters respectively, followed by locating the point of intersection.

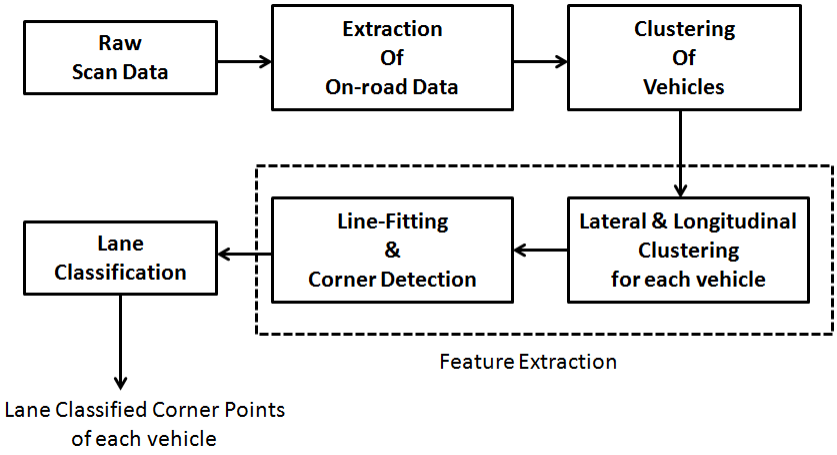


Fig. 3.3.: Stages of vehicle detection.

The following gives a detailed description of each of the stages involved in the process of vehicle detection:

1. **Clustering of detected objects**: This step involves assigning a geometric shape to the clusters of vehicles on the road. The data obtained comprises of scan-points pertaining to all the vehicles on the road. The scan-points are divided into clusters, taking the Euclidean distance between two consecutive scan-points into account.

   The segmentation criterion [53] considers two consecutive points d_k and d_{k+1} (as observed by laser-scanner), belonging to the same cluster if the distance between them fulfils the following expression:

$$|d_k - d_{k+1}| = d_{min}\frac{\tan(\beta)\sqrt{2(1-\cos(\phi)}}{\cos(\phi/2)-\sin(\phi/2)} + c_1 \tag{3.6}$$

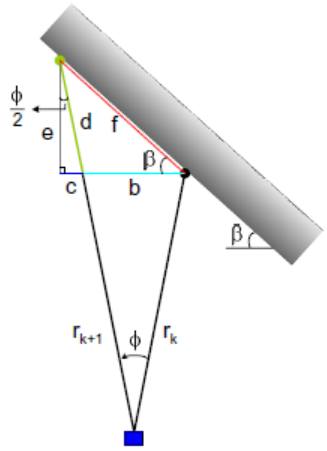
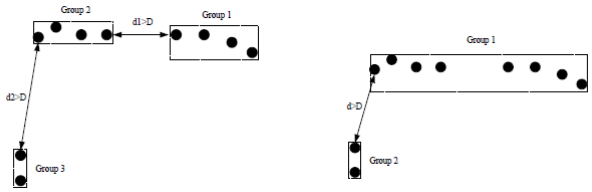


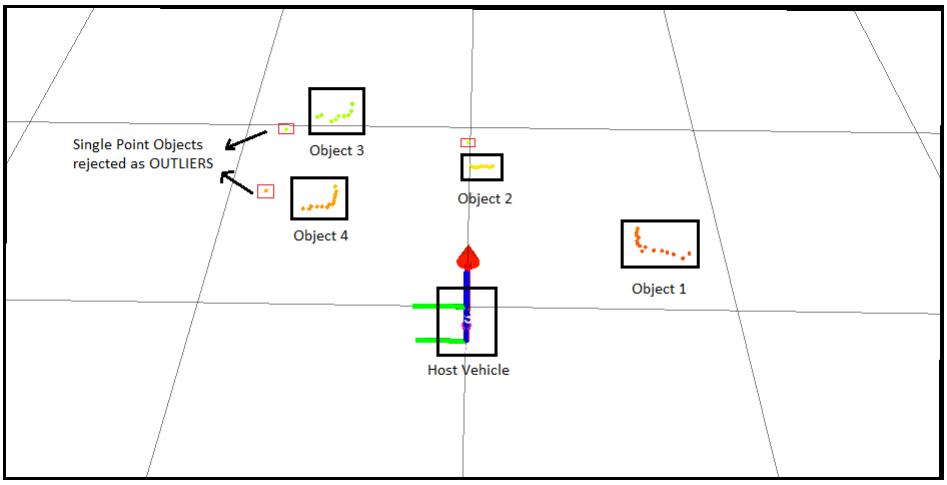
Fig. 3.4.: Clustering of data points into respective vehicles [53].

where $d_{min} = \min\{d_k, d_{k+1}\}$, ϕ is the angular resolution of the laser-scanner, β is the parameter that reduces the dependence on the measured range, c_1 is the threshold that handles the longitudinal error of the sensor. Figure 3.5 demonstrates another simple criterion based on distance threshold to cluster different obstacles [17].

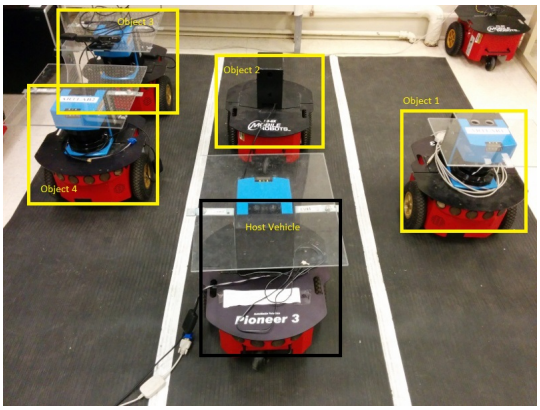


(a) Grouping of 3 obstacles. (b) Grouping of 2 obstacles.

Fig. 3.5.: Significance of distance threshold [17] .



(a) Clustering of 4 obstacles.



(b) Real World Scenario.

Fig. 3.6.: Clustering of detected objects.

2. **Road-data segmentation**: This segment deals with the identification of information that plays a significant role in target-tracking of neighboring vehicles. It involves extraction of the road and lane information from the surroundings. This is performed using some of the commonly used camera-based algorithms [54]. A brief overview of such algorithms and their role in vehicle detection are discussed as follows:

   Interaction with camera-based algorithm: The method of vehicle segmentation using a 2D camera [55] is reliable in terms of accuracy. It can be used to perform a number of tasks as follows:

   (a) Road and associated data-extraction: It involves segmentation of the road-data along with the vehicles on the road. Color-based techniques and feature-based techniques are used to detect the road segment.

   (b) Lane detection: This involves detection of lanes, and hence calculation of the number of lanes. Based on the width of each detected lane, we estimate the overall width of the road.

   (c) Lane localization: This method localizes the host vehicle within its lane by indicating its relative position from the boundary of the lane.

   With the lane width w_L, road width w_R and number of lanes N_L, we can prepare a road model. With the lane number of the host vehicle l_N and its relative position l_P, we can localize the position of the host vehicle and also determine the direction of traffic flow in the adjacent lanes. This algorithm helps in the classification of the detected vehicles into their respective lanes that creates a well-structured dataset of target vehicles useful during data-association (discussed in Chapter 4).

3. **Feature-extraction**: As the vehicles move in and out of the sensor-frame, the part of the vehicle observed by the scanner changes at every iteration. To handle this shape-change problem, we have proposed a method to track linear

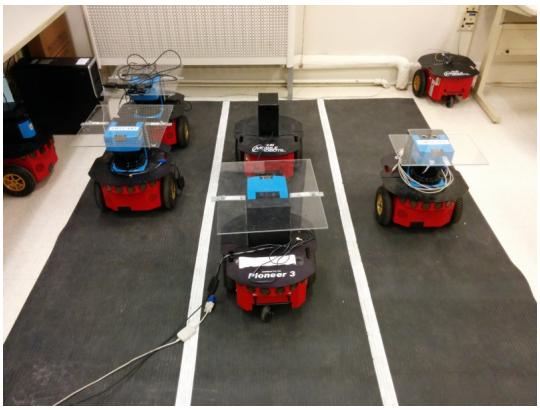


Fig. 3.7.: Real world scenario (camera).

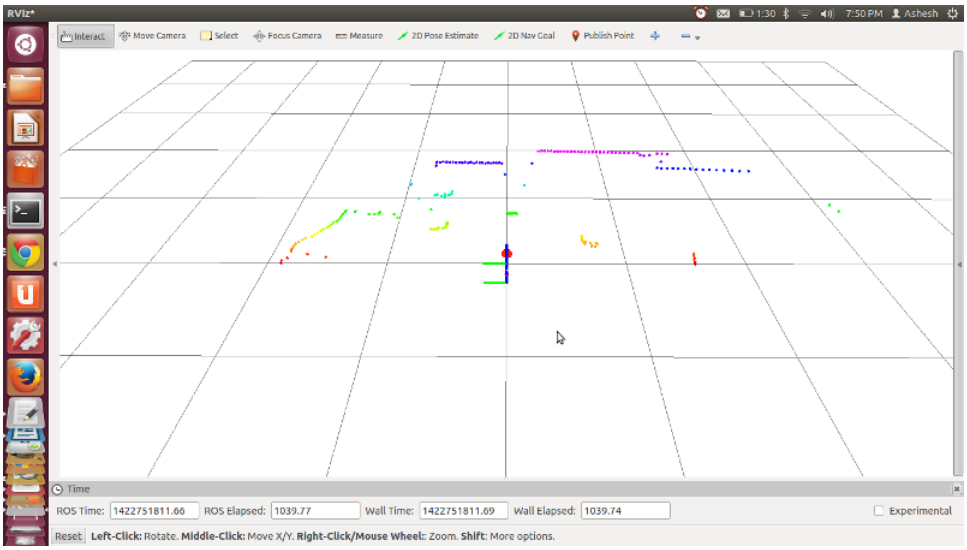


Fig. 3.8.: Unsegmented laser-scan.

features (lines and corners) of the scan-data. Major advantages of this feature tracking are as follows:

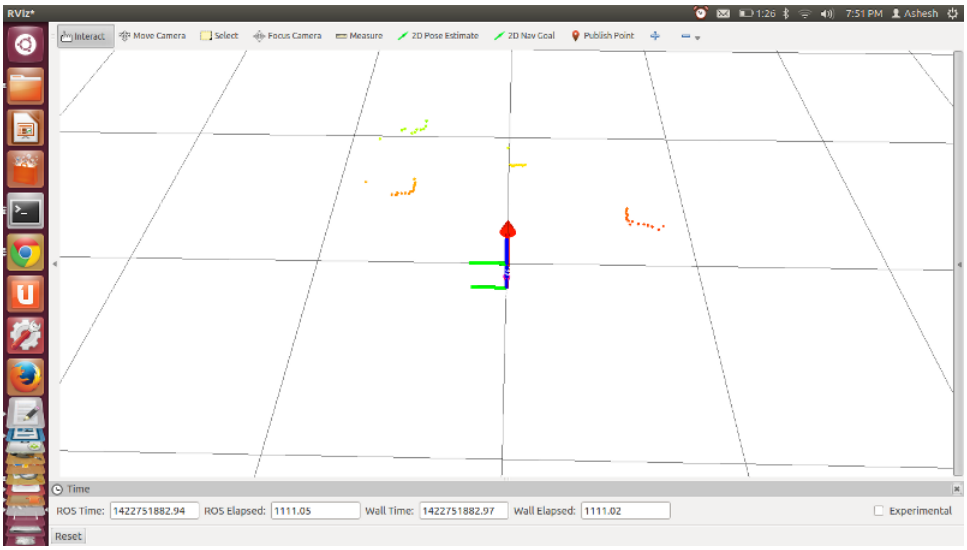


Fig. 3.9.: Segmented road data (scan).

(a) Corners are stable features not affected by shape change.

(b) It is an efficient method with the scan-points being reduced to a few significant features.

(c) A good rectangle corner fit is evidence that an object is a vehicle, not a bush or ground return.

A typical approach towards feature-extraction using linear features is seen in [56]. A more detailed research regarding feature-extraction has been done in [57] to perform shape-invariant feature-extraction dependent on the intrinsic sensor characteristics [58]. As mentioned, this stage comprises of two important tasks namely, clustering of the scan-points of each vehicles into lateral and longitudinal groups, and corner detection using the intersection of the lines fit through these clusters of points. If a corner is not visible, then the mid-point of the visible linear feature is used for vehicle detection. The stages involved in the proposed corner detection for scatter plots is as follows:

(a) **Clustering**: This is performed using a variation of the Ramer-Douglas-Pecker poly-line simplification algorithm [15]. It considers a line between the first and last points in a set of points that form the curve. It checks which point is at the farthest distance from the constructed line. If the point is at a distance closer than a given threshold ϵ, it removes all the points those in between. If, on the other hand the outlier point is at a distance greater than ϵ, the curve is split into two parts. The algorithm is simplified by considering the maximum distance d_{max} from the perpendicular distances of each points, to the line connecting the initial and final points. If the corner is visible, we use it as the extracted feature for correspondence matching to the future measurements.

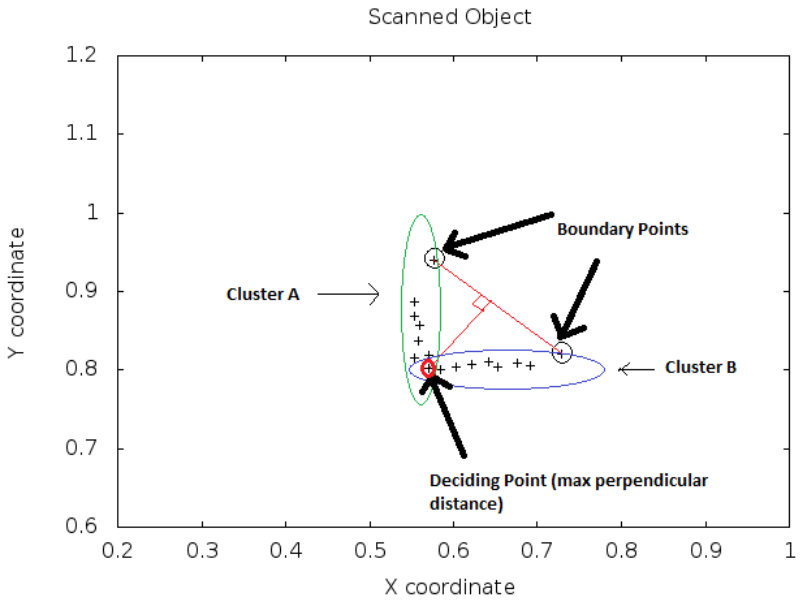


Fig. 3.10.: Clustering of points for one vehicle.

(b) **Corner detection**: Once the scan-points are clustered into lateral and longitudinal categories, linear regression is performed on the two sets to get

a linear approximation of the two near-perpendicular clusters. However, the most common problem associated with linear regression is the influence of outliers on the inlier set. A single outlier can potentially threaten the accuracy of the entire linear approximation. In order to overcome this problem, the data is passed through RANSAC (Random sample consensus) based linear-regression algorithm.

Random sample consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers [59]. As it is a non-deterministic algorithm, it produces results with a certain probability that increases with increased number of iterations. A good accuracy is achieved by implementing the RANSAC-based linear regression. Linear regression is an approach for modeling the relationship between a scalar dependent variable Y, and one or more explanatory variables denoted by X [60]. We model the linear relationship between the scan-points for individual data-set to a fit a straight line that approximates the distribution according to:

$$Y = aX + b \tag{3.7}$$

where

$$a = \frac{n\Sigma(xy) - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2}$$

$$b = \bar{Y} - b\bar{X}$$

and $\bar{X}$ and $\bar{Y}$ denote the mean values of the respective coordinate data and n is the number of scan-points in each cluster.

We find the intersection of the two lines (approximately perpendicular) obtained for each cluster. The point of intersection is the corner that can be used as a significant feature for correspondence matching at every iteration. However, from an algorithmic perspective, solving equations of lines as cross product of the coefficients greatly reduces the computation time.

Let us assume the two lines are $a_1x + b_1y + c_1 = 0$ and $a_2x + b_2y + c_2 = 0$, we can form two vectors

$$L_1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} \qquad L_2 = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \tag{3.8}$$

$$P' = L_1 X L_2 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.9}$$

where P' is in the homogeneous form and hence can be converted to a 2D coordinate form as

$$P = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \tag{3.10}$$

where P denotes the extracted point of intersection.

Algorithm 1 Modified Douglas-Pecker Corner Detection Algorithm for 2D points.

Construct a line between initial and final points.
Compute perpendicular distances to the line $\{dp_j\}$, $j = 1, 2, .., N$ scan-points.
$N^* = \underset{j}{\text{argmax}}\ (dp_j)_{j=1}^{N}$
$d_{max} = dp_{N^*}$
if ($d_{max} >$ threshold):
    form 2 clusters about point N^* (including N^* in both clusters)
    perform RANSAC-based line-fitting on each cluster to obtain L_1 & L_2
    $P = L_1 \times L_2$ (intersection of the lines) yields the corner point,
else:
    form only one cluster
    perform RANSAC line-fit operation on the cluster
    mid-point of the line gives the feature point P.

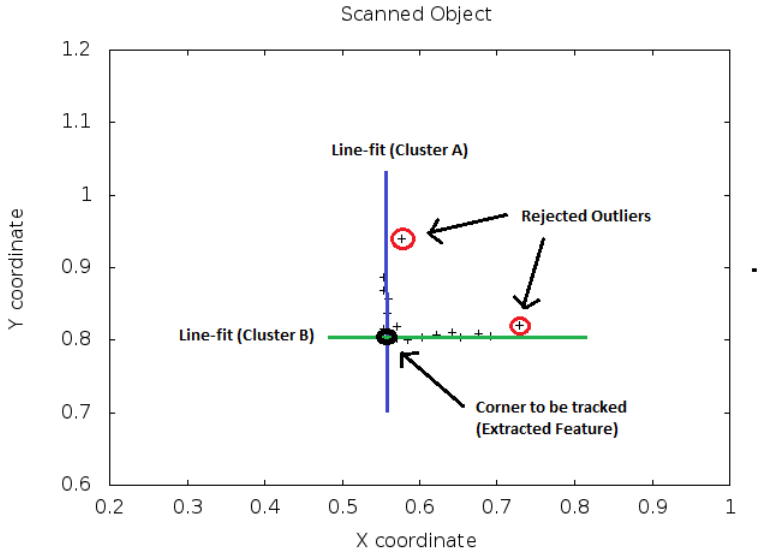
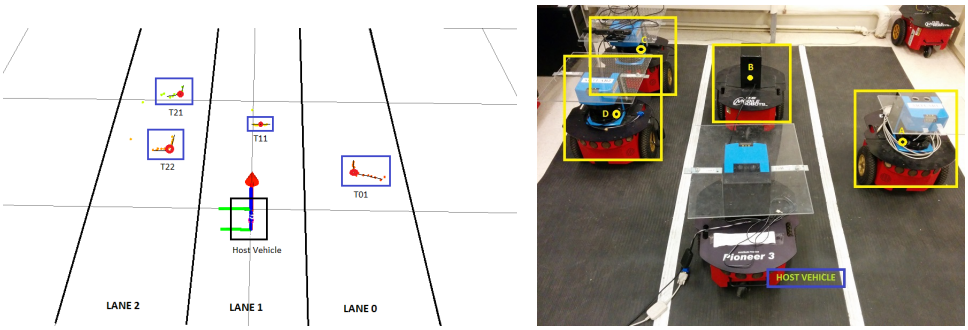


Fig. 3.11.: Corner detection using modified Douglas-Pecker method.



(a) Lane classified vehicle data. (b) Real world scenario.

Fig. 3.12.: Lane classified data from vehicle detection algorithm.

3.4 Summary

This chapter provides an overview of laser-scanning techniques, its advantages and its application to target-tracking. It discusses about sensing of the environment, detection of vehicles from the environment and performing feature-extraction of the target vehicles. A modified Douglas-Pecker poly-line simplification algorithm has been proposed to perform corner extraction of the target vehicles from scatter-points obtained using the laser-scanner sensor. Corner detection has been achieved with an average of 1.5cm accuracy using SICK LMS-200 laser-scanner in a simulated lane environment. The feature-extraction algorithm has been integrated with available camera-based lane-detection and localization algorithms to perform lane-classification of the detected vehicles. A well-structured dataset is obtained as a result of the vehicle detection algorithm that improves the computation time and complexity of the tracking algorithm. The position and lane information associated with each target vehicles, as provided by the detection algorithm is utilized extensively during target-tracking, discussed in the following chapter (Chapter 4).

4. INTERACTIVE-MULTIPLE-MODEL-BASED TWO-STAGE VELOCITY-ESTIMATOR

4.1 Introduction to target-tracking

This Chapter focuses on tracking of vehicles in the neighborhood of the host vehicle and estimating their velocities and angular turn-rates with respect to the host vehicle. target-tracking refers to the problem of using sensor measurements to determine the location and path of the objects of interest. A number of sources of uncertainty in the object-tracking problem renders it a highly non-trivial task. Such uncertainties involve object motion being subject to random disturbances, objects going undetected by sensors, number of objects in sensor-frame changing randomly, measurements being subject to noise, objects being close to each other with overlapping measurements, etc. This chapter discusses the different stages involved in the design of a two-stage estimator for tracking maneuvering targets. The first stage involves the design of a turn-rate estimator using an adaptive-window-based method. The second stage performs data association of the measurements to the target vehicles followed by velocity estimation using a standard Kalman filter. After a detailed analysis of each stage, the complete Interactive-Multiple-Model-based (IMM) two-stage estimator will be discussed in detail with an overview of some of the standard filters implemented during the process. Simulations and experimentation have been carried out for both the stages as well as for the maneuvering multiple target-tracking scenarios to validate the performance of the proposed velocity-estimator.

4.2 Target and sensor-space models

This section performs modeling of the target vehicle kinematics and sensors involved in target-tracking. In Section 4.2.1, a general overview of state-space model used for modeling vehicle kinematics and sensor characteristics will be described. Section 4.2.2 will provide information regarding two possible models to characterize the motion of the target vehicles. In Section 4.2.3, the model used for extracting sensor information will be discussed.

4.2.1 State-space model

For tracking a moving target, it is necessary to extract information about the target state from sensor measurements. As information regarding vehicle kinematics and sensor characteristics are generally known, most tracking algorithms use predefined mathematical model of the target to improve their performance. The problem of describing the target motion model requires a compromise between accuracy and complexity. The following state-space model describes the state and measurement functions:

$$\begin{aligned} X_{k+1} &= F_k X_k + G_k u_k + E_k \nu_k \ , \\ z_k &= H_k X_k + w_k \ , \end{aligned} \tag{4.1}$$

where $X_k \epsilon \mathbb{R}^n$, $z_k \epsilon \mathbb{R}^m$ and $u_k \epsilon \mathbb{R}^p$ are the target state, observation, and control input vectors, respectively, at the discrete-time t_k ; $\nu_k \epsilon \mathbb{R}^q$ and $w_k \epsilon \mathbb{R}^m$ are the process and measurement noise sequences with covariance matrices $Q(t) \epsilon \mathbb{R}^{q \times q}$ and $R(t) \epsilon \mathbb{R}^{m \times n}$ respectively; and $F_k \epsilon \mathbb{R}^{n \times n}$, $G_k \epsilon \mathbb{R}^{n \times p}$, $E_k \epsilon \mathbb{R}^{n \times q}$ and $H_k \epsilon \mathbb{R}^{m \times n}$ are the discrete-time transition matrix, input gain, noise gain and measurement matrix, respectively.

The discrete-time model (Eq. (4.1)) is obtained by discretizing the continuous-time model:

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) + D(t)\nu(t) \ ; \quad x(t_0) = x_0 \ , \\ z(t) &= C(t)x(t) + w(t) \ , \end{aligned} \tag{4.2}$$

where $A(t), B(t), D(t)$ and $C(t)$ are the continuos-time equivalent to the F_k, G_K, E_k and H_k matrixes respectively, and $\nu(t)$ and $w(t)$ are the continuous-time process and measurement noise with covariances $Q(t)$ and $R(t)$, respectively.

4.2.2 Mathematical model for target vehicles

The concept of using noise-driven system dynamics is to consider the fact that noise with different levels of variance can represent different motions. For example, a model with high variance noise can be used to depict maneuvering motion (lane-changing scenario), while a model with low variance noise represents uniform motion (adaptive cruise control). Considering the normal and tangential accelerations as a_n and a_t, respectively, we can implement various kinematic models [61] to capture the complex maneuvering behavior (rectilinear and curvilinear motions) of the target vehicles:

1. **Constant-velocity model**: Assuming that steady-state accelerations are quite small, linear accelerations or decelerations can be represented by process noises with the constant-velocity model; that is, the constant-velocity model along with a zero-mean noise and an appropriate covariance representing the magnitude of acceleration can characterize uniform-velocity motion on the road (i.e., $a_n = 0, a_t = 0$) [62] . In discrete-time formulation, the constant-velocity model with noise is given by

$$X(k+1) = F_{CV}X(k) + \Gamma\nu(k) \ , \tag{4.3}$$

where

$$F_{CV} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & T \end{bmatrix} \quad \text{and} \quad \Gamma = \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix}$$

and $X(k) = [x \ v_x \ y \ v_y]^T$ is the state-vector, where x, y denote positions and v_x, v_y denote velocities in the x-y plane, respectively, at k^{th} iteration with sampling

time T, ν is a zero-mean Gaussian white noise representing accelerations with covariance Q where

$$Q = E[\Gamma\nu(k)\nu^T(k)\Gamma^T] = \Gamma\sigma_\nu^2\Gamma^T$$
$$= \sigma_\nu^2 \begin{bmatrix} T^4/4 & T^3/2 & 0 & 0 \\ T^3/2 & T^2 & 0 & 0 \\ 0 & 0 & T^4/4 & T^3/2 \\ 0 & 0 & T^3/2 & T^2 \end{bmatrix} , \tag{4.4}$$

where $E[.]$ denotes the expectation function, and Γ^T denotes the transpose of Γ, and σ_ν denotes the variance.

2. **Rectilinear accelerated motion model**: The assumption of treating the longitudinal accelerations as noise might not be applicable in real-world scenarios. During the scenarios where $a_n = 0$ but $a_t \neq 0$, it is essential to append acceleration into the state-vector for better accuracy of the model in capturing the motion of the target vehicle. This updated model expressed in discrete-time directly is given by

$$X(k+1) = F_A X(k) + \Gamma\nu(k) \ , \tag{4.5}$$

where

$$F_A = \begin{bmatrix} F^* & 0_{3\times3} \\ 0_{3\times3} & F^* \end{bmatrix} \qquad \Gamma = \begin{bmatrix} \Gamma^* & 0_{3\times1} \\ 0_{3\times1} & \Gamma^* \end{bmatrix}$$
$$F^* = \begin{bmatrix} 1 & T & 0 \\ 0 & T & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \Gamma^* = \begin{bmatrix} T^2/2 \\ T \\ 1 \end{bmatrix}$$

and $X(k) = [x \quad v_x \quad a_x \quad y \quad v_y \quad a_y]^T$ is the state-vector with a_x and a_y as the acceleration in the x-y plane with sampling time T, and ν is a zero-mean Gaussian white noise representing jerks with covariance Q given by

$$Q = E[\Gamma\nu(k)\nu(k)^T\Gamma^T] = \Gamma\sigma_\nu^2\Gamma^T$$
$$= \sigma_\nu^2 \begin{bmatrix} Q^* & 0_{3\times3} \\ 0_{3\times3} & Q^* \end{bmatrix} , \tag{4.6}$$

where

$$Q^* = \begin{bmatrix} T^4/4 & T^3/2 & T^2/2 \\ T^3/2 & T^2/2 & T \\ T^2/2 & T & 1 \end{bmatrix} \ .$$

3. **Coordinate turn-rate model**: A discrete-time model for curvilinear motion is derived from a continuous-time model for coordinated turn motion [63]. A turn with a constant yaw rate along a road of constant radius of curvature $[a_n \neq 0, a_t = 0]$ is referred to as a constant-speed turn. Noise is added to a constant-speed turn model for the purpose of capturing variations in the road curvature. The noise representing the modeling error can be in the form of angular acceleration or non-constant radius of curvature. For a vehicle turning with a constant angular rate and moving with constant-speed, the kinematic equations in the x-y plane are

$$\dot{y}(t) = \omega x(t) \tag{4.7}$$

where $\dot{y}(t)$ denotes the lateral velocity, and ω is the constant yaw rate (positive ω implying a counter-clockwise turn). The tangential component of the acceleration is equal to the rate of change of the speed:

$$\ddot{y}(t) = \frac{d\dot{y}(t)}{dt} = \frac{d(\omega x(t))}{dt} = \omega \dot{x}(t) \tag{4.8}$$

where $\dot{x}(t)$ denotes the longitudinal velocity. The normal component is defined as the square of the speed in the tangential direction divided by the radius of the curvature of the path:

$$\ddot{x}(t) = -\frac{\dot{y}^2(t)}{x(t)} = \frac{\omega^2 x^2(t)}{x(t)} = \omega \dot{y}(t) \ . \tag{4.9}$$

The state-space representation of the above system with the state-vector defined by $X(t) = [x(t) \quad \dot{x}(t) \quad y(t) \quad \dot{y}(t)]^T$ is:

$$\dot{X}(t) = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & -\omega \\ 0 & 0 & 0 & 1 \\ 0 & \omega & 0 & 0 \end{bmatrix} X(t) + \nu(t)$$

$$\dot{X}(t) = F_{CT} X(t) + \nu(t) \ , \tag{4.10}$$

where $\nu(t)$ represents zero mean, white Gaussian process noise. For a time-invariant continuous-time system with sampling rate T using Euler discretization, we obtain

$$F_k \triangleq F(T_{k+1}, T_k) = F(T_{k+1} - T_k) = F(\Delta T) = e^{A\Delta T} \ . \tag{4.11}$$

. The discretized process noise w_k retains the same characteristics

$$\begin{aligned} E[w_k] = 0 \quad , \quad E[w_k w_j] = Q_k \delta_{kj} \\ Q_k = \int_0^{\Delta T} e^{A(\Delta T - \tau)} B Q B^T e^{A^T (\Delta T - \tau) d\tau} \ . \end{aligned} \tag{4.12}$$

Performing the Euler discretization on the continuous system Eq. (4.2), we obtain,

$$e^{F_{CT}T} = \begin{bmatrix} 1 & \frac{\sin(\omega(k)T)}{\omega(k)} & 0 & -\frac{1-\cos(\omega(k)T)}{\omega(k)} \\ 0 & \cos(\omega(k)T) & 0 & -\sin(\omega(k)T) \\ 0 & \frac{1-\cos(\omega(k)T)}{\omega(k)} & 1 & \frac{\sin(\omega(k)T)}{\omega(k)} \\ 0 & \sin(\omega(k)T) & 0 & \cos(\omega(k)T) \end{bmatrix} X(k) \ . \tag{4.13}$$

It has been remarked that if the angular rate ω in Eq. (4.10) is time-varying, Eq. (4.13) would be no longer true. Hence, a nearly constant-speed turn model in a discrete-time domain is introduced. In this approach, the model is motivated from Eq. (4.10), but the angular rate is allowed to vary. This concept is

extended to the case where the angular rate varies abruptly but can be estimated using other means and substituted in the state-space model. The nearly constant-speed turn model is defined as follows:

$$X(k+1) = \begin{bmatrix} 1 & \frac{\sin(\omega(k)T)}{\omega(k)} & 0 & -\frac{1-\cos(\omega(k)T)}{\omega(k)} \\ 0 & \cos(\omega(k)T) & 0 & -\sin(\omega(k)T) \\ 0 & \frac{1-\cos(\omega(k)T)}{\omega(k)} & 1 & \frac{\sin(\omega(k)T)}{\omega(k)} \\ 0 & \sin(\omega(k)T) & 0 & \cos(\omega(k)T) \end{bmatrix} X(k) + \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix} \nu(k) \tag{4.14}$$

where the state-vector $X(k) = [x \;\; v_x \;\; y \;\; v_y]^T$. For the proposed algorithm, the road curvature is assumed to be minimal and a small amount of noise is added for capturing variations in curvature. The turn-rate is assumed to be piecewise constant, switching to unknown values at unknown time, but remaining constant between the switching intervals. Also, it has been assumed that the the turn-rate values occur within an interval $[-\omega_{max},\ \omega_{max}]$. The proposed algorithm for curvilinear velocity estimation requires 5 such Coordinate Turn-rate (CT) models, details to be discussed later in this section. By adding acceleration dimension a_x and a_y into state-vector [64], [65], $X(k) = [x \;\; v_x \;\; a_x \;\; y \;\; v_y \;\; a_y]^T$, the state models get modified as:

$$F_{CTA} = \begin{bmatrix} F_{new} & 0 \\ 0 & F_{new} \end{bmatrix} \qquad \gamma = \begin{bmatrix} \gamma_{new} & 0 \\ 0 & \gamma_{new} \end{bmatrix} \tag{4.15}$$

where

$$F_{new} = \begin{bmatrix} 1 & \frac{\sin(\omega(k)T)}{\omega(k)} & \frac{1-\cos(\omega(k)T)}{\omega(k)^2} \\ 0 & \cos(\omega(k)T) & \frac{\sin(\omega(k)T)}{\omega(k)} \\ 0 & -\omega(k)\sin(\omega(k)T) & \cos(\omega(k)T) \end{bmatrix} \text{ and } \gamma_{new} = \begin{bmatrix} 0.167T^3 \\ 0.5T^2 \\ T \end{bmatrix}. \tag{4.16}$$

4. **Fixed-structure, variable-structure and augmented models**: As mentioned, some of the common models used for vehicle tracking are constant-velocity model, rectilinear-acceleration model, coordinated turn-rate model, etc. Each of these models can be categorized into fixed-structure, variable-structure or augmented models.

   (a) **Fixed-structure models:** Such models are not modified throughout the execution of the algorithm; that is, the structure and components of the state-space model is independent of time.

   (b) **Variable-structure models:** Such models vary during the execution of the algorithm. The difference may be in terms of the governing equations or the values of the model parameters. The proposed algorithm characterizes the system in terms of variable models dependent on time-varying turn-rate parameter.

   (c) **Augmented models:** Such type of modifications in a state-space model is to account for an additional parameter in the state-space; that is, the parameters involved in the governing equations that were previously not a part of the state-space can be augmented into the state-space, thus increasing the dimension of the state-space vector. For example, in the case of coordinate turn-rate model, we can augment the turn-rate ω as a new parameter into the state-space vector, $X(k) = [x \quad v_x \quad y \quad v_y \quad \omega]^T$.

4.2.3 Mathematical model for sensory system (measurement model)

To extract maximum information about the target vehicle from the measurements, modeling the sensor is important. The sensor used is a 2D laser-scanner and placed at an origin of the Cartesian coordinate system; that is, placed at the coordinates $(x, y) = (0, 0)$. The sensor measurements are in polar coordinate system, providing range r and azimuth angle θ .

The sensor measurements are modeled as:

$$r = \bar{r} + v_r \tag{4.17}$$

$$\theta = \bar{\theta} + v_\theta \tag{4.18}$$

where $\bar{r}$ and $\bar{\theta}$ represent the nominal target position in polar coordinates, and $v_k = [v_r, v_\theta]$ represents the respective measurement errors, assumed to be zero-mean, Gaussian distributed and uncorrelated. With v_k as the measurement error vector in polar coordinates at time step k, we obtain the distribution of error as an approximate Gaussian:

$$v_k \sim N(0, R_k^p) \tag{4.19}$$

with

$$R_k = cov(v_k) = diag(\sigma_r^2, \sigma_\theta^2) \quad \forall k \tag{4.20}$$

where R_k denotes the measurement covariance matrix with σ_r^2 and σ_θ^2 representing the range covariance and angle covariance, respectively. The sensor measurements in the sensor coordinate system is given by

$$z_k^p = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix} + v_k \tag{4.21}$$

At this point, there are two distinct coordinate systems, the Cartesian coordinate system of the target and the polar coordinate system of the sensor. In this research, Cartesian coordinates is used for sensor data representation and hence requires coordinate transformation. The corresponding Cartesian coordinates of the target position is given by:

$$x = r\sin\theta \quad , \quad y = r\cos\theta \ . \tag{4.22}$$

After conversion, the sensor model in Cartesian coordinates is given as:

$$z_k = HX_k + w_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} X_k + w_k \tag{4.23}$$

where w_k is the measurement noise associated with the 2D laser-scanner (zero-mean Gaussian) in Cartesian coordinates with covariance described as:

$$R = E[w(k)w(k)^T] = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma_{yy} \end{bmatrix} \quad , \tag{4.24}$$

where R represents the measurement covariance matrix.

The advantage of a linear form of output model is that the Kalman filter [66] can be applied for tracking purposes. The challenge is to obtain a linearized conversion of the measurement noise, since v_k might be non-Gaussian and coupled across coordinates. Techniques to track non-linearity present only in the measurement noise and not in the measurement function is not a trivial problem and needs extensive research. In this research, a simplified linearized model (i.e., Eq. (4.23)) has been used as an approximate output model.

4.3 Tracking of maneuvering targets

4.3.1 Introduction

Various attempts at performing tracking of maneuvering targets have been made. The method in [25] assumes known turn-rate, which is unrealistic in a highway scenario. Tracking performance would deteriorate when the assumed turn-rate differs significantly from the true one. Another approach involves multiple models in an IMM environment with all possible turn-rates taken into consideration [26] where the increased number of filters result in sub-optimal performance. To minimize the number of filters, methods with turn-rate augmented into the state-vector have been used for tracking curvilinear motion [27], sometimes integrating cross-track acceleration as system input [28]. This difficult nonlinear problem requires computationally expensive estimators like Intelligent Kalman Filter (IKF) [29], Unscented Kalman Filter (UKF) [30] or even Particle Filters (PF) [16]. Another approach is to separately estimate the turn-rate (adaptive turn-rate estimation [31]) and then proceed with the state-vector update [32]. In this research, a method of tracking a maneuvering target

has been proposed that uses a combination of variable- and fixed-structure models, and an adaptive-window-based technique for estimating the turn-rate independently followed by velocity estimation.

4.3.2 Adaptive-window-based method for curvilinear velocity estimation

The adaptive-window-based method is an extension of the Adaptive-Grid (AG) method [45] for tracking a maneuvering target undergoing motion with unknown turn-rate (ω). The turn-rates are assumed to be lying within a particular interval $[-\omega_{max},\ \omega_{max}]$. Comprising of 3 variable-structure (VS) and 2 fixed-structure (FS) models, this 5 model Interactive Multiple Model (IMM) algorithm estimates the position, velocity and turn-rate of the target vehicles during both linear and curvilinear motions. Figure 4.1 gives an overview of the adaptive-window-based method of turn-rate estimation.

The 5 models form a coarse window at the beginning with

$$\omega = [\omega_{LS} \quad \omega_L \quad \omega_C \quad \omega_R \quad \omega_{RS}] \ . \tag{4.25}$$

The model with turn-rate ω_C estimates the turn-rate of the target vehicle. The two other VS models with turn-rates ω_L and ω_R, respectively, bounds the converging turn-rate space; that is, this determines the width of the adaptive-window that converges towards the true turn-rate. The 2 FS models (ω_{LS} and ω_{RS}) remain constant and indicate the extreme values of turn-rate in both positive and negative directions. They are utilized in shifting the converged space towards the new turn-rate value during abrupt changes in ω.

At each cycle, the true turn-rate is determined by the probabilistic weighted sum of all the turn-rates:

$$\omega_C^{k+1} = \mu_{LS}^k \omega_{LS}^k + \mu_L^k \omega_L^k + \mu_C^k \omega_C^k + \mu_R^k \omega_R^k + \mu_{RS}^k \omega_{RS}^k \ , \tag{4.26}$$

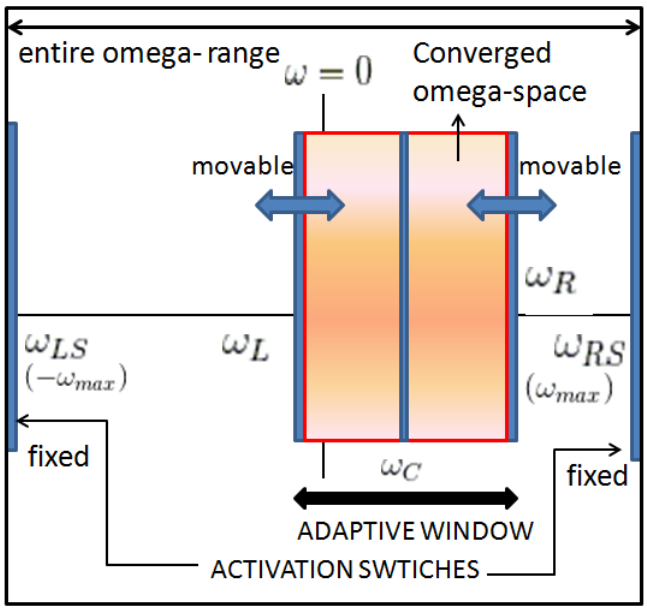


Fig. 4.1.: Turn-rate (ω) convergence.

where $\mu = [\mu_{LS} \quad \mu_L \quad \mu_C \quad \mu_R \quad \mu_{RS}]^T$ denotes the mode probability matrix that get updated based on the outcome of IMM filter estimates. Once the turn-rate is estimated, the turn-rates of the variable-structure models are updated accordingly:

$$\lambda_L^K = \|\omega_C^{k+1} - \omega_L^k\| \quad \text{and} \quad \lambda_R^K = \|\omega_C^{k+1} - \omega_R^k\| \tag{4.27}$$

$$\omega_L^{k+1} = \begin{cases} \max\{\omega_C^{k+1} - \max\{\frac{3}{4}\lambda_L^k, \delta\omega\}, \omega_{LS}\} \ \mu_L < p_1 \ \text{ or } \ \mu_L > p_2 \\ \max\{\omega_C^{k+1} - \max\{\frac{1}{4}\lambda_L^k, \delta\omega\}, \omega_{LS}\} \ p_1 < \mu_L < p_2 \end{cases} \tag{4.28}$$

$$\omega_R^{k+1} = \begin{cases} \min\{\omega_C^{k+1} - \max\{\frac{3}{4}\lambda_R^k, \delta\omega\}, \omega_{RS}\} \ \mu_R < p_1 \ \text{ or } \ \mu_R > p_2 \\ \min\{\omega_C^{k+1} - \max\{\frac{1}{4}\lambda_R^k, \delta\omega\}, \omega_{RS}\} \ p_1 < \mu_R < p_2 \end{cases} \tag{4.29}$$

where ω_C^{k+1} denotes the estimated turn-rates at $(k+1)^{th}$ iteration, λ_L^K and λ_R^K are the distance of the previous boundaries at k^{th} iteration, $\delta\omega$ is a measure of the minimum width of the omega space, p_1 and p_2 indicate the confidence for selecting the validity

of the models. The parameters $\delta\omega$, p_1, and p_2 are design parameters and can be adjusted based on the performance.

The number of steps for calculating the respective variable turn-rates (Eq. (4.28) and Eq. (4.29)) are lesser than the existing AG method. The proposed algorithm allows a faster shift towards the estimated turn-rate when a high model selection confidence exists. It helps in ensuring that turn-rate converges to the true value within a maximum of 3-4 seconds during abrupt changes. Furthermore, the two fixed models that mark the boundaries of the turn-rate become more probable during abrupt changes in ω and drive the window towards the new value of turn-rate.

Role of curvilinear velocity estimation in autonomous navigation

The proposed algorithm for turn-rate estimation as mentioned in this section, is the first stage of the proposed two-stage estimator for multi-target-tracking of vehicles on the road. It has been validated for scenarios involving a maneuvering target and also for highway with simulation of multiple vehicles on the road. An example of a trajectory undertaken by a maneuvering target is shown in Fig. 4.2. The turn-rate tracking performed by a stationary sensor and subsequent linear velocity tracking results is shown in Figs. 4.3 and 4.4. In this case, a stationary sensor with a sampling rate of 20Hz observes the target for a period of $700s$. As observed (Figs. 4.3 and 4.4), the target can be tracked with great accuracy, and the locking time of turn-rates during abrupt changes is quite low (about 2-3 second).

A comparison of the performance of various maneuvering target-tracking algorithms has been shown in Table 4.1. Fixed-Structure IMM (FS-IMM) and constant-speed IMM (CS-IMM) fail to track highly maneuvering targets unless ω values are known and represented with appropriate models. Augmented IMM (AIMM) works for all ω values but is computationally expensive due to non-linearity. The Adaptive-Grid IMM (AG-IMM) works for a narrow ω range with a large locking delay during abrupt ω changes. The proposed Adaptive-Window IMM (AW-IMM) performs bet-

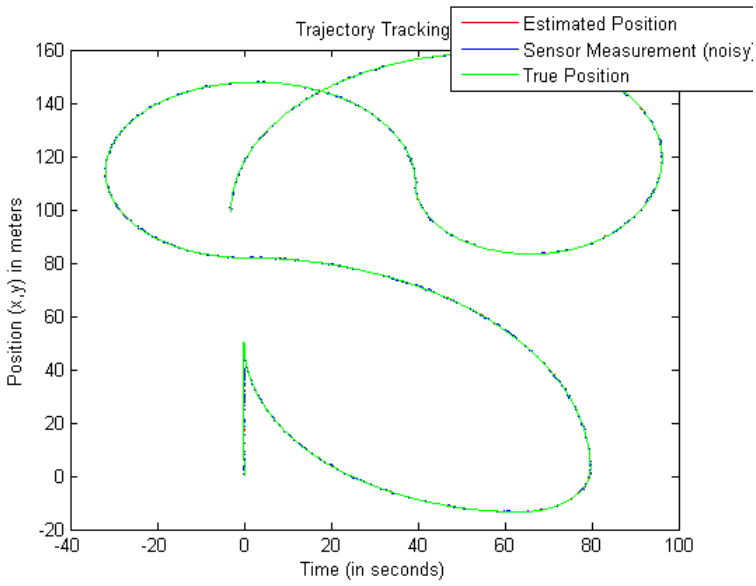


Fig. 4.2.: Trajectory for maneuver target-tracking using stationary sensor.

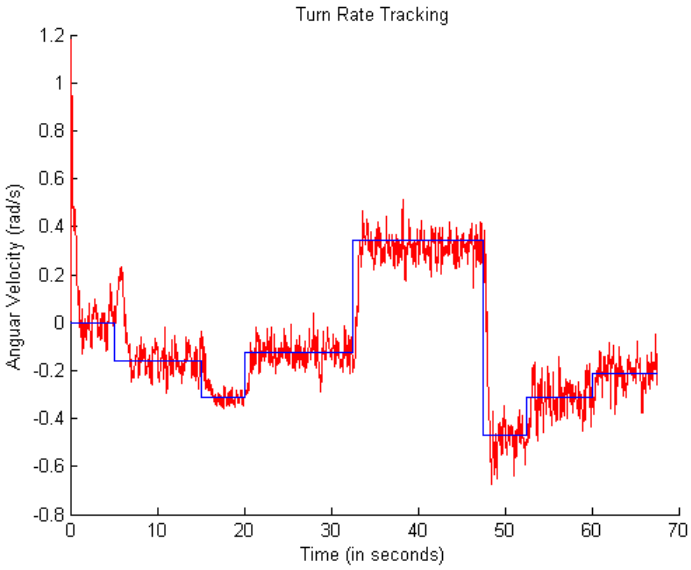


Fig. 4.3.: Turn-Rate tracking for stationary sensor scenario.

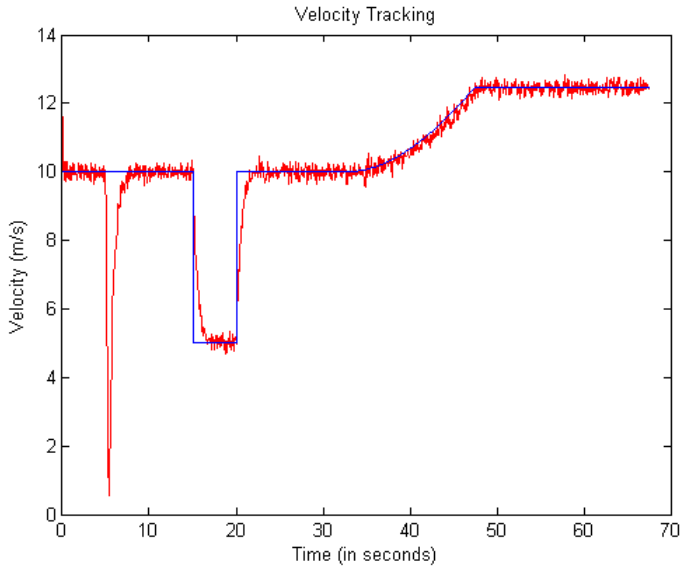


Fig. 4.4.: Linear velocity tracking for stationary sensor scenario.

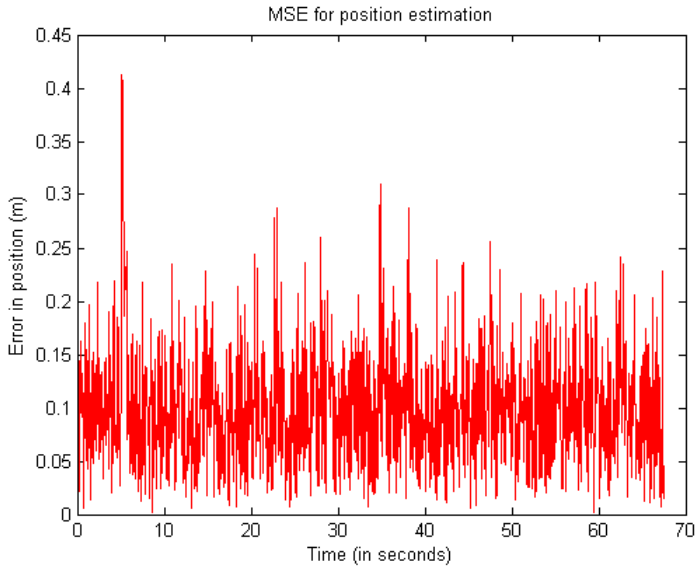


Fig. 4.5.: Mean-squared position error for stationary sensor scenario.

Table 4.1: Comparison of maneuvering target-tracking algorithms.

Algorithm	$\omega = 0$	$\omega \neq 0$	**Large range**	**Abrupt ω change**
CS-IMM	Valid	$\omega \approx 0$	not valid	not valid
FS-IMM	Valid	known ω	many models	locking delay
AIMM	Valid	valid	one model	delay $> 5s - 6s$
AG-IMM	Valid	valid	only 3 models	delay $< 5s - 6s$
AW-IMM	Valid	valid	only 5 models	delay $< 4s$

ter by eliminating non-linear computations and restricting the locking delay to 3-4 seconds.

The proposed algorithm is then applied to simulations of certain autonomous navigation scenarios. A significant example of curvilinear trajectory is that of a lane-changing scenario. Two such significant lane-changing scenarios have been used for simulation:

1. **Target vehicles undergoing lane-change**: This scenario depicts target vehicles undergoing lane-change with the host vehicle moving straight ahead. The simulation scene in Fig. 4.6 describes a 3-lane environment with the host vehicle in the center lane moving with a constant-speed of 4 m/s (linear motion).

   Each target vehicle moves at 6 m/s with two of the vehicles in linear motion and the other two performing a lane-change. The turn-rate estimation for the respective target vehicles is shown in Fig. 4.7. As observed, locking of turn-rate up to 10% accuracy is achieved within 3-4 seconds after the initiation of the lane-change manuever. Figures 4.8 and 4.9 demonstrate the velocity tracking and mean-squared position error of the target vehicles. The host vehicle estimates positions and velocities of target vehicles with root-mean-squared error up to 5 cm - 6 cm and 0.25 m/s - 0.3 m/s ($3.5\% - 4\%$), respectively.

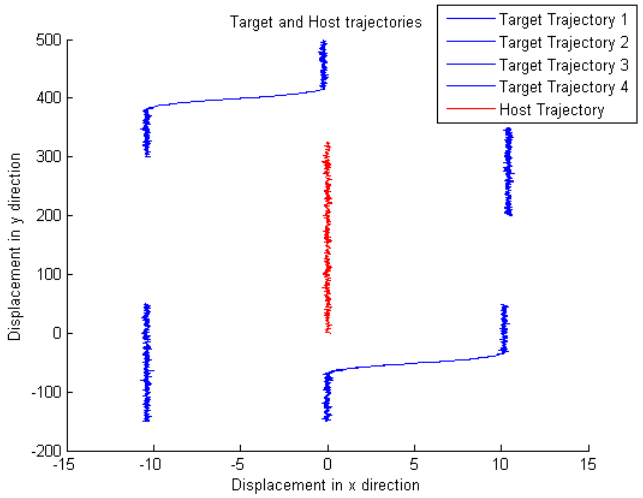


Fig. 4.6.: Scenario with target vehicles undergoing lane-change.

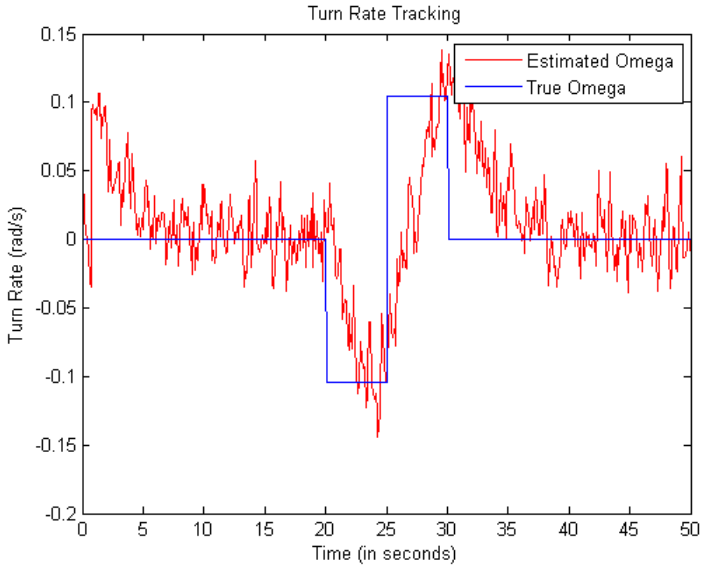
2. **Host vehicle undergoing lane-change**: This scene involves the host vehicle undergoing a lane-change. The proposed algorithm can track target vehicles moving straight ahead as well as undergoing lane-change when the host vehicle is also performing a lane-change.

In Fig. 4.10, a scenario is set up with the host vehicle undergoing a lane-change. Tracking results are shown for the target vehicles in linear (Fig. 4.12) and curvilinear motion (Fig. 4.11). Although the accuracy of turn-rate estimation drops as compared to the case of host vehicle in linear motion, the performance of the velocity and position tracker is not affected.

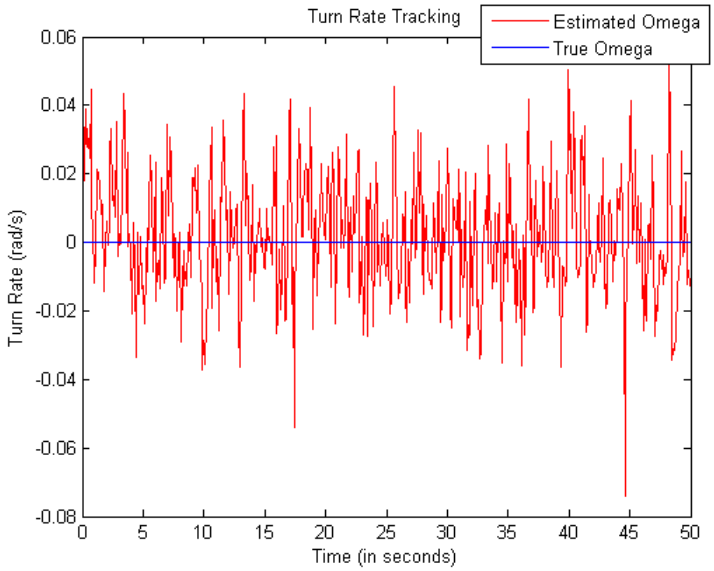
4.4 Data association and multiple target-tracking

4.4.1 Introduction

By combining data association concept with state estimation algorithms, the accuracy of multi-target-tracking can be improved as compared to applying estimation

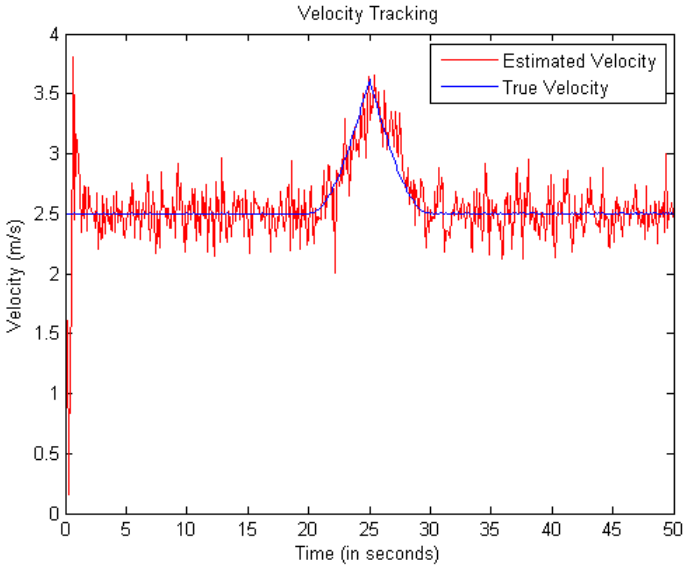


(a) Target undergoing curvilinear motion.

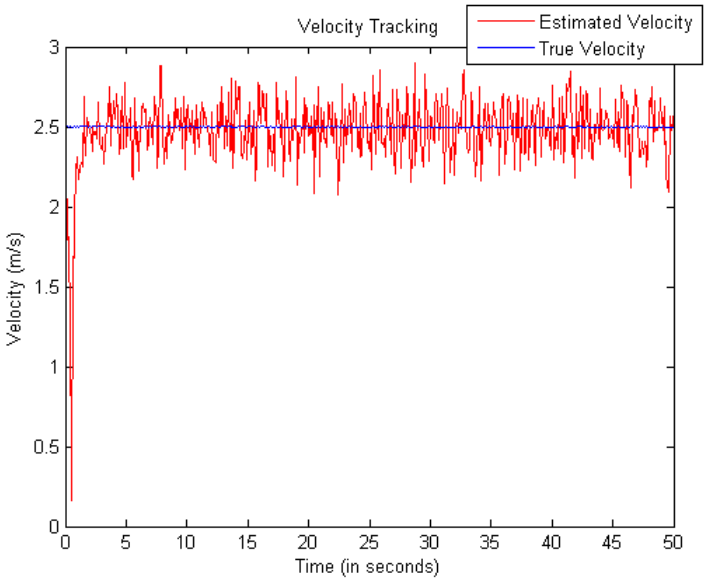


(b) Target undergoing linear motion.

Fig. 4.7.: Turn-Rate tracking of target vehicles.

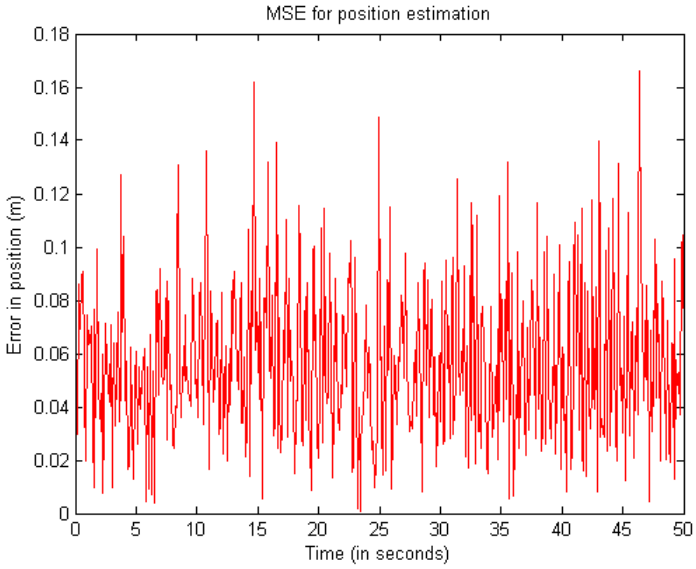


(a) Target undergoing curvilinear motion.

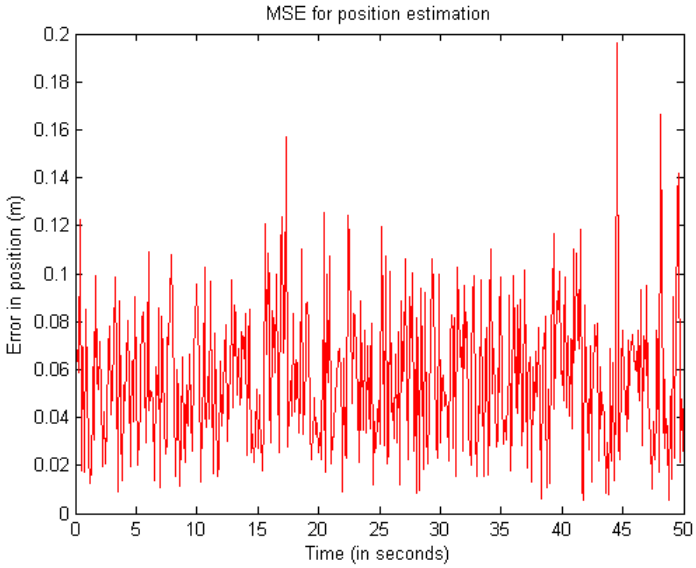


(b) Target undergoing linear motion.

Fig. 4.8.: Velocity tracking of target vehicles.



(a) Target undergoing curvilinear motion.



(b) Target undergoing linear motion.

Fig. 4.9.: Mean squared error (MSE) for position tracking.

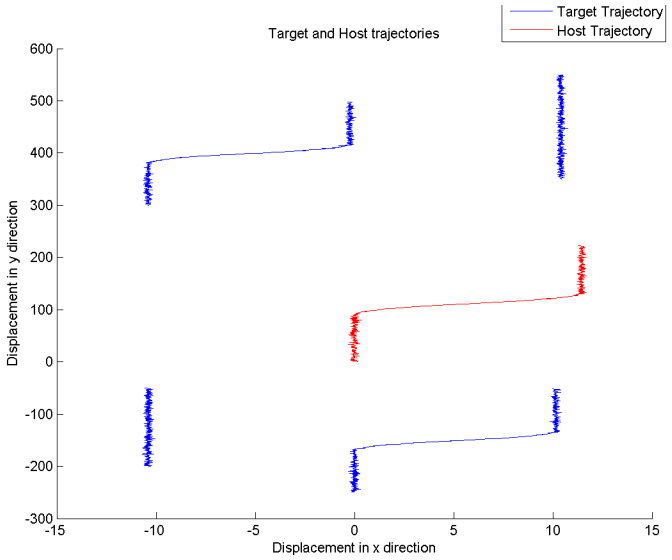
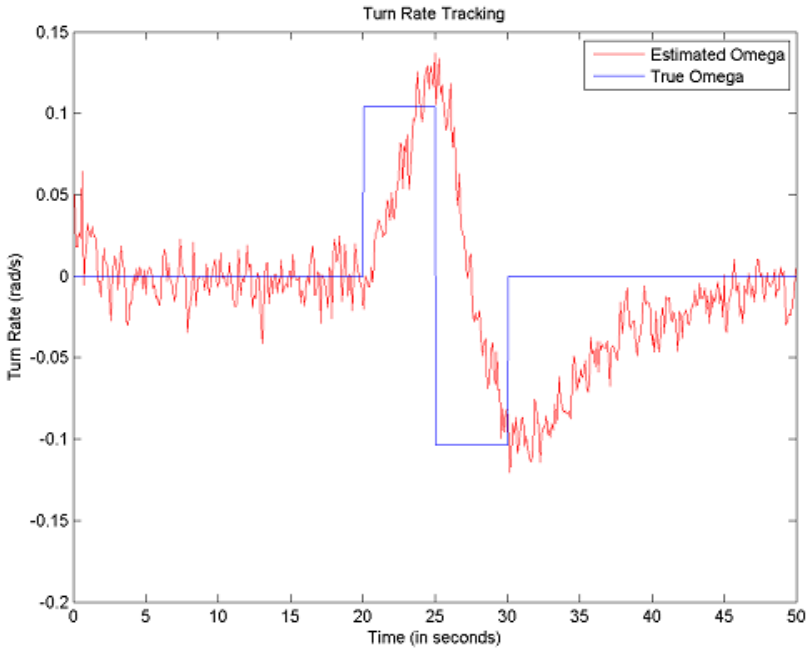
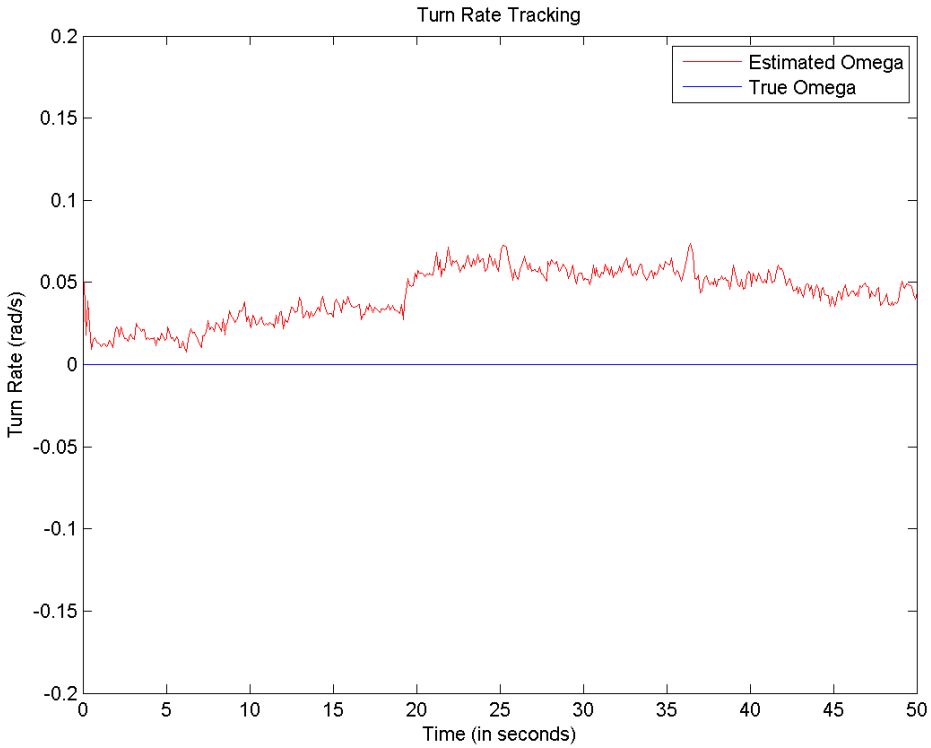


Fig. 4.10.: Scenario with both host vehicle and target vehicles undergoing lane-change.

procedure to the combined state-space model. Among the various methods, Neares-Neighbor (NN) filter [67] and Joint Probabilistic Data Association Filter (JPDAF) algorithm [34] are the most commonly used. Althought NN filter was extensively used initially, over the years it has paved way to different variations of JPDAF for better performance of tracking systems [8], in particular to robotics and autonomous systems [68]. Instead of common multi-tracking problems of data overlap and co-alescence of cluttered data, situations involving continuously changing dataset are faced in a highway scenario with vehicles moving in and out of the sensor frame. During such scenarios, association of the correct measurements with the appropri-ate vehicles become important. In this research, a variant of the traditional Nearest Neighbor-JPDAF (NN-JPDAF) method has been proposed for the well structured lane-classified vehicle data. Unlike most of the existing multiple target-tracking algo-rithms, the proposed method updates the object's position and velocity individually once each valid pairing takes place.

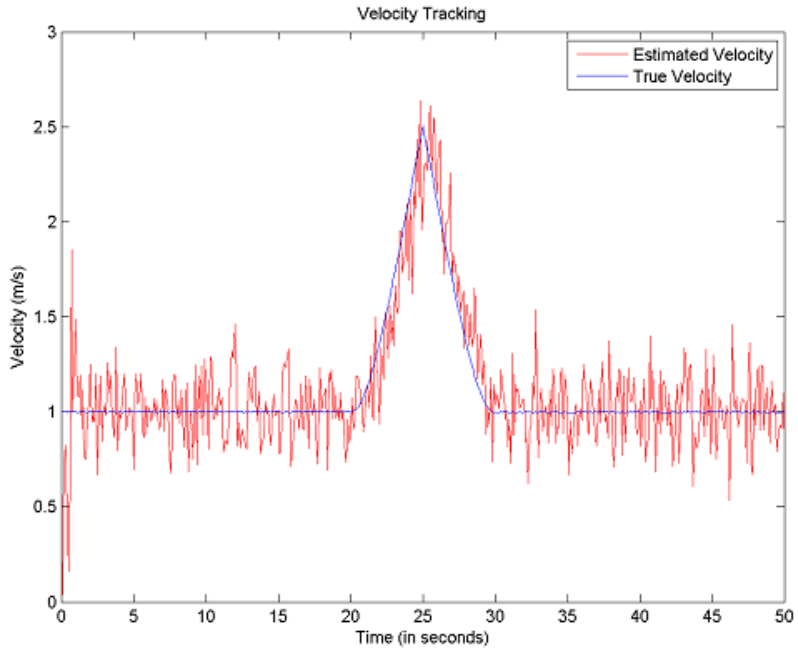


(a) Target undergoing curvilinear motion.

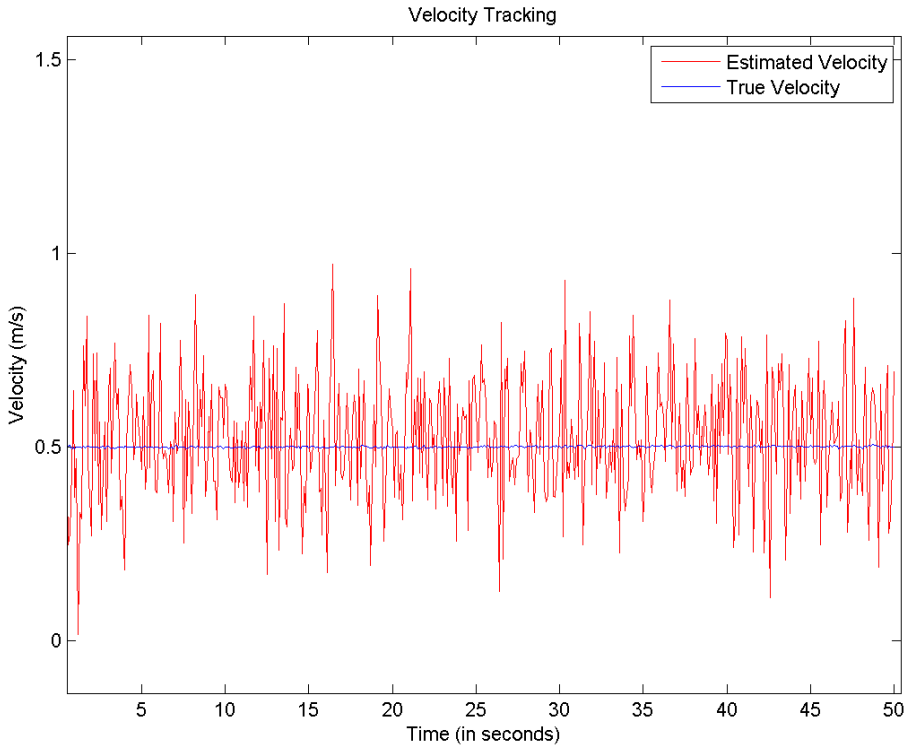


(b) Target undergoing linear motion.

Fig. 4.11.: Turn-rate tracking of target vehicles.

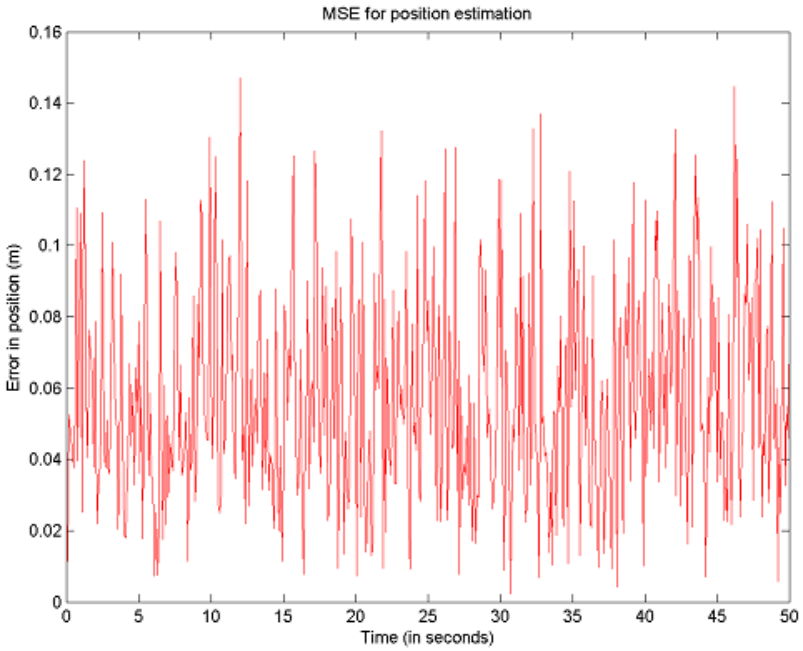


(a) Target undergoing curvilinear motion.

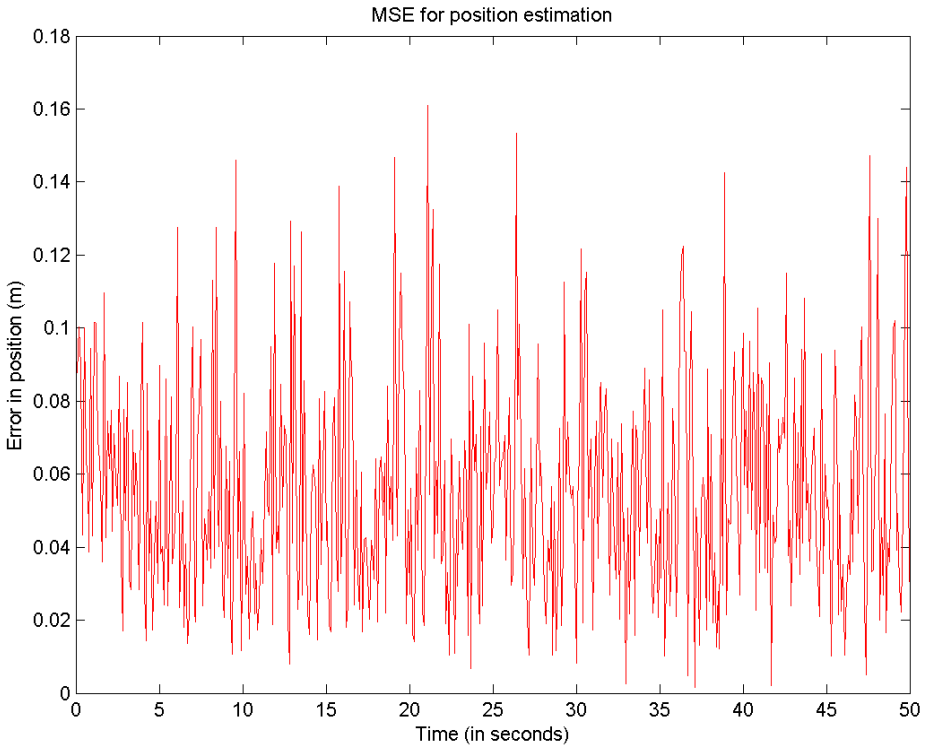


(b) Target undergoing linear motion.

Fig. 4.12.: Velocity tracking of target vehicles.



(a) Target undergoing curvilinear motion.



(b) Target undergoing linear motion.

Fig. 4.13.: Mean squared error (MSE) for position tracking.

4.4.2 Need for data association

1. Clutter Scenario: A major concern during target-tracking when multiple vehicles are in close proximity is the overlapping of their measurements. Measurements of unknown origin occur where it becomes difficult to associate them with the appropriate target vehicles. In Fig. 4.14, measurement M_2 is of uncertain origin; that is, it can be associated with T_1, T_2 or it could be a noisy invalid measurement.

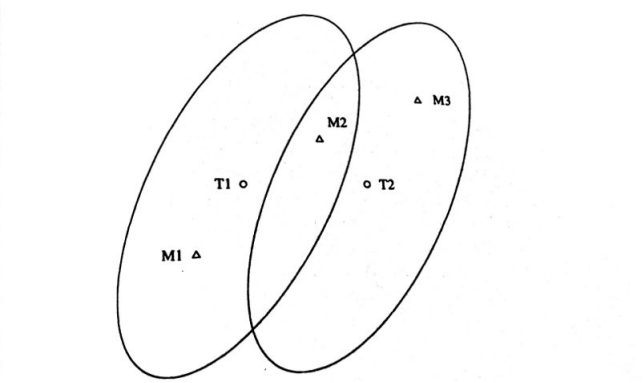


Fig. 4.14.: Validation regions for two targets involving uncertainty [69].

2. Dynamic Dataset Scenario: Another concern during target-tracking is the continuously changing dataset with the targets moving in and out of the sensor-frame. It results in the loss of accuracy and efficiency if a measurement is incorrectly associated with the target. Examples to demonstrate such scenarios have been discussed in Chapter 2 (Fig. 2.1). As a result of mis-associations, it would take a few iterations for the tracking algorithm to regain its accuracy. However, this can be avoided by correctly updating the state-vector table at every iteration, taking care of the vehicles added and removed from the scenario.

4.4.3 Comparison of Nearest Neighbor and Joint Probability Data Association filters

As mentioned earlier, the two most popular approaches to the problem of data association are Nearest-Neighbor filters and Joint Probability Data Association filters. A brief insight into each of the algorithms will bring out the basic difference between the two approaches towards data association.

1. **Nearest Neighbor Standard Filter (NNSF)**: It selects the measurement nearest to the predicted measurement (in terms of Euclidean distance) as the only validated measurement and rejects all the remaining observations [67]. This method assumes high probability of detection for the targets and a low rate of clutter. The most common problems associated with this approach are overconfident variances and filter divergence as the nearest measurement might not always be the correctly associated measurement [70].

2. **Joint Probability Data Association Filter (JPDAF)**: This approach integrates all the measurements in the proximity of the expected measurement and performs the update of association pairs based on the association probability of each measurement [34]. It improves tracking in regions of high clutter densities where NNSF becomes unreliable.

To avoid some of the shortcomings of JPDAF, a combination of both the data association methods, known as Nearest Neighbor Joint Probability Association Filter (NN-JPDAF) has been formulated by Fitzergald [71], in which the weighted average updating of tracks is replaced by one-to-one assignments of measurements to targets using association probabilities.

4.4.4 Modified nearest neighbor joint probability data association filter based multiple target-tracking

This subsection provides an overview of the multiple target-tracking of the vehicles in the neighbourhood of the host vehicle. A modified approach to the existing Nearest-Neighbor-based JPDAF algorithm [72] has been proposed to simplify the data association for lane-classified data. The following stages provide a mathematical approach to the proposed solution of the data-association problem:

Stage 1: Validated target-measurement pair

The validation region for each target vehicle is computed, and based on which, the potential target-measurement pairs are generated. The validation region is constructed with the predicted measurement $\hat{Z}(k|k-1)$ so that the target-oriented measurement falling in this region has a probability of P_G. Validation region can also serve in creating a conflict zone around the target vehicle helpful for collision avoidance [73].

Algorithm 2 Valid target-measurement pair.

1: **for** t=1:total number of vehicles
2: **for** j=1:total number of measurements
3: $\nu = \hat{Z}(k|k-1) - F^{*}_{\mu_{max}} \hat{x}_{(i,j)}(k)$
4: $d^2_{tj} = \nu^T S^{-1}_{(i,j)} \nu$
5: **if** $(d^2_{tj} < g^2)$
6: record the $(i-j)$ pair

In algorithm 2, S is measurement prediction covariance corresponding to the mode with highest probability, d^2_{tj} is the Euclidean distance to measurements, $F^{*}_{\mu_{max}}$ is the state transition matrix corresponding to the most probable mode, and g is the threshold for ensuring that the target oriented measurement falls in the validation region with a probability of P_G. A valid association is said to occur if the normalized distance from the target's predicted measurements (d^2) is less than g^2 (chi-distribution),

Stage 2: Likelihood and association probability

Calculation of the likelihood and subsequently the association probability of the measurements with the targets is performed assuming a normal distribution.

$$\Lambda_{cj} = \frac{1}{\sqrt{|2\pi S_{cj}|}} exp\{-0.5\boldsymbol{\nu}_{c,j}^T S_{c,j}^{-1} \boldsymbol{\nu}_{c,j}\} \tag{4.30}$$

where Λ_{cj} denotes the likelihood of the j^{th} measurement to be associated with the c^{th} target. With q models in the IMM filter, we choose the likelihood corresponding to the most probable mode $\boldsymbol{\mu}_{max}$

$$\Lambda_{cj} = \Lambda_{cj}^{(q)}|_{q->\boldsymbol{\mu}_{max}} \ . \tag{4.31}$$

Association probability gives the sense of how likely the target-measurement pair is to be associated. The association probability (β_{cj}) of target c associating with measurement j [72] is approximated as:

$$\beta_{cj} = \frac{\Lambda_{cj}}{T_c + M_j - \Lambda_{cj} + \epsilon} \ , \tag{4.32}$$

where the quantity T_c is the sum of all likelihoods for target c, which is given by

$$T_c = \sum_{j=1}^{m} \boldsymbol{\nu}_{cj} \ ,$$

and M_j, which is the sum of all likelihoods for measurement j, is

$$M_j = \sum_{c=1}^{N} \boldsymbol{\nu}_{cj} \ .$$

and ϵ is a bias that accounts for non-unity probability of detection.

Stage 3: Finding correct association pairs

This stage of the algorithm finds the best target-measurement pairs for each valid measurement from the available pairs (T^* targets and M^* measurements) by choosing the ones with the highest association probability:

$$[c^*, j^*] = \max((\beta_{cj})_{j=1}^{T^*})_{m=1}^{M^*} \tag{4.33}$$

where $[c^*, j^*]$ indicates the best target,measurement pair. This is a technique similar to the standard nearest-neighbor method, except that association probabilities are used instead of the normalized distances and a search is made for the largest probability instead of the smallest normalized distance.

Stage 4: Insertion and deletion of vehicles

Based on a validation matrix Eq. (4.34), we can eliminate target vehicles without any measurements as vehicles that have moved out of the sensor frame. Also, the measurements that are not associated with any vehicle are considered as new vehicles. For example, let's say that the there exists targets $1, 2, 3, 4, 5$ mapped to the measurements $1, 2, 3, 4, 5$ in a way $1- > 1, 2- > 2, 3- > 4, 4- > 5$, respectively, then the validation matrix V is:

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

where the columns represent the target vehicles and the rows denote the measurements. A zero column corresponds to a removed target and a zero row denotes a potential new measurement. A full rank validation matrix indicates complete data association with no change in the surroundings.

Stage 5: Update state-vectors

The final stage performs the state-vector estimation individually for the correctly associated target-measurement pairs. We perform updates for $(i - c^i)$ pair for $i = 0, 1, 2, \ldots, m$ where m denotes the number of valid measurements and c^i denotes the corresponding target vehicle.

Algorithm 3 provides a layout of the modified NNJPDAF algorithm performed for each lanes during multi-target-tracking. The first part updates and deletes existing

targets based on association probability. The measurements still unaccounted for, are then assigned as new target vehicles in the second part of the algorithm.

Algorithm 3 Multiple target-tracking.

for j = 1 : no. of target vehicles
    for i = 1 : measurements in the same lane
        compute $d^2_{(j,i)} = \boldsymbol{\nu}^T S^{-1} \boldsymbol{\nu}$
for c = 1 : no. of validated target vehicles M^*
    for m = 1 : validated measurements
        compute likelihoods $\Lambda_{(c,m)}$
        compute association $\beta_{(c,m)}$
$[\beta_{max}, c^*, m^*] = \underset{(c,m)}{max}((\beta_{c,m})_{c=1}^{T^*})_{m=1}^{M^*}$
if $(\beta_{max} > threshold_{validation})$:
    update state-vector for (c^*, m^*) pair
    remove (c^*, m^*) pair from analysis
else:
    delete jth target
    update total number of targets
****
for k=1 : number of measurements
    if (M^k exists)
        insert target at k^{th} location
        update total number of targets

4.4.5 Experimentation using Pioneer-3DX mobile robot

The experimentation comprises of the host vehicle (a P3-DX mobile robot) moving in the center lane of a 3-lane road, undergoing a linear motion with uniform velocity. A 180-degree field-of-view 2D laser-scanner has been used, hence the vehicles behind the host vehicle are outside the sensor-frame. The scene begins with two target

vehicles in Lane-2 (C_{22} and C_{21}) and one in Lane-0 (C_{01}) visible within the sensor-frame. The layout of the vehicle nodes are shown in Fig. 4.15, where A, B, C, and D indicate the target vehicles. After time interval t_1, the scene comprises of 2 target

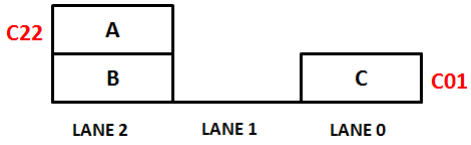


Fig. 4.15.: Stage 1: Layout of the target nodes.

vehicles, where C_{22} gets updated to C_{21} with C_{01} remaining the same as before. The algorithm ensures correct measurement association with the existing target vehicles (see Fig. 4.16). Similarly, after time interval t_2, it is observed that a new vehicle

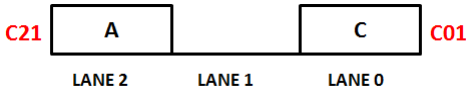


Fig. 4.16.: Stage 2: Layout of the target nodes.

enters into the frame, C_{22} and C_{01} exit the frame from the rear. The state-vector table gets further updated with this latest information (Fig. 4.17). After the entire simulation time T, the scene (Fig. 4.18) comprises of only one target vehicle in Lane-2 (C_{21}) and the host vehicle in the center-lane.

During the course of the entire simulation, there are three vehicles A, B and C that get deleted from the sensor-frame. Also, there are two switches in the vehicle label that occur, A from C_{22} to C_{21} and D from C_{22} to C_{21}. These two changes would not be detected soon unless the state-vector table is updated at every iteration.

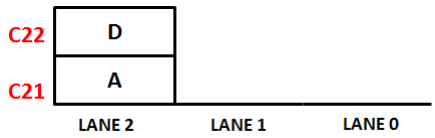


Fig. 4.17.: Stage 3: Layout of the target nodes.

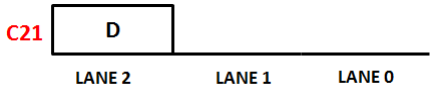


Fig. 4.18.: Stage 4: Layout of the target nodes.

As far as each of the target vehicle information is concerned, the data is stored using a linked-list representation. As seen in Fig. 4.19, each of the target vehicle

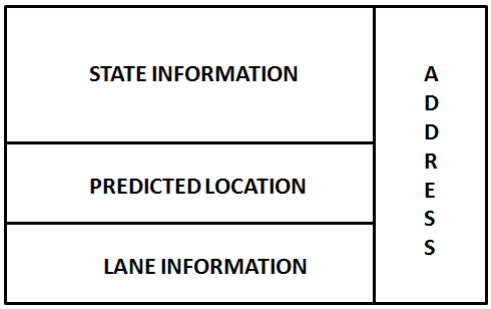


Fig. 4.19.: Structure of a single target node.

node consists information about the state, the predicted measurement as well as the associated lane information. With N vehicles in operation, the worst time complexity would be linear $O(N)$ for addition and removal of target vehicles.

Since the number of target vehicles is not fixed, it is difficult to validate the performance of the algorithm based on a standard error metric. Hence, we utilize two slightly different metrics to measure the effectiveness of data association algorithm: the normalized correct associations (NCA) and incorrect-to-correct association ratio (ICAR) [74]:

$$\text{NCA} = \frac{\text{number of correct associations}}{\text{number of total associations}}$$

$$\text{ICAR} = \frac{\text{number of incorrect associations}}{\text{number of correct associations}}$$

The objective of any multi-target-tracking algorithm is to attain high NCA and low ICAR. Ideally, a value of 1 is desirable for the NCA parameter indicating correct data associations for all valid measurements. ICAR should be typically close to 0 but due to noise and clutter environment, a higher value might be obtained (false new vehicles).

As shown in Fig. 4.20, the ICAR performs quite well with a slight drop in performance when both the host vehicle and the target vehicles are undergoing lane-change simultaneously. The simulations demonstrate the worst-case scenarios, where around 1-2 unknown objects are treated as new vehicles or are mis-classified (during lane-change maneuver due to continuously changing orientation of the sensor) and hence, result in loss of accuracy.

As shown in Fig. 4.21, NCA performs quite well due to a well-structured lane-classified data maintained for each target vehicle. However its accuracy drops when both the host vehicle and target vehicles undergo lane-change simultaneously; that is, better associations are made during linear motion of the host vehicle.

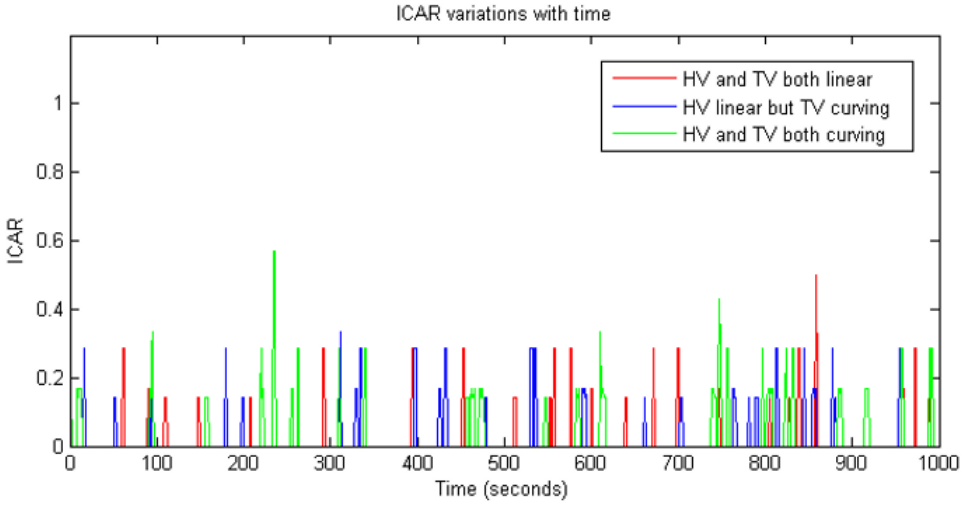


Fig. 4.20.: ICAR variations over time.

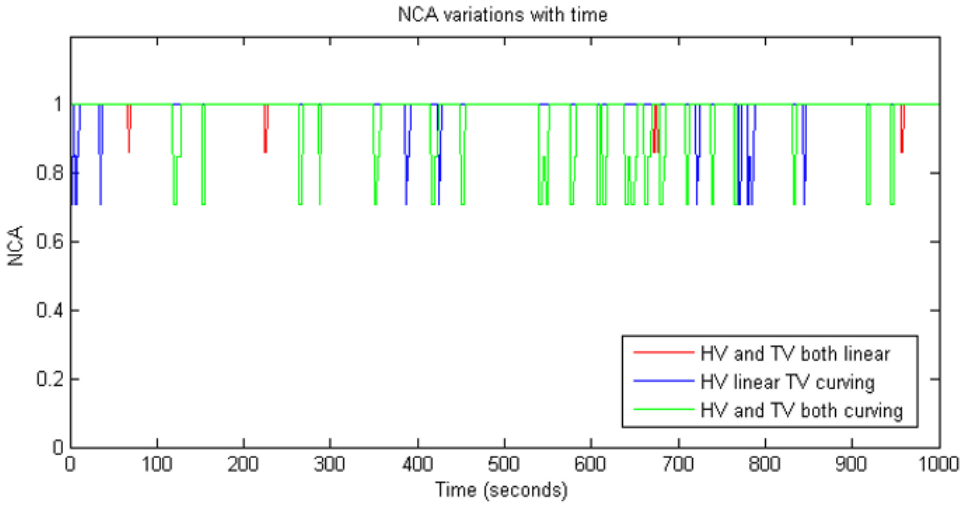
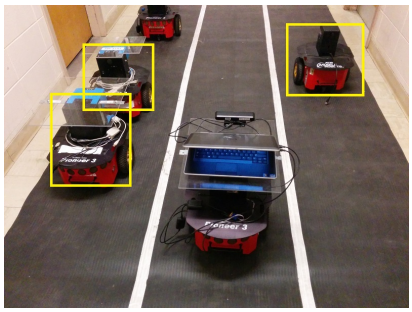
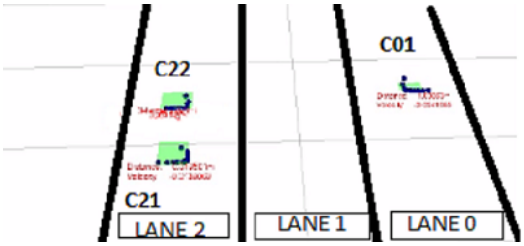


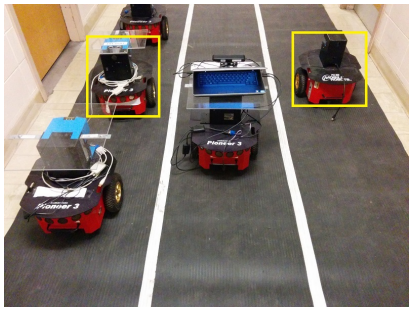
Fig. 4.21.: NCA variations over time.



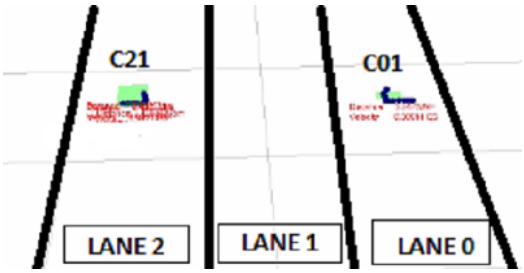
(a) Stage1 (camera).



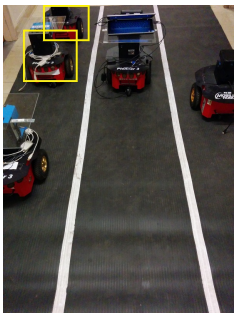
(b) Stage1 (laser).



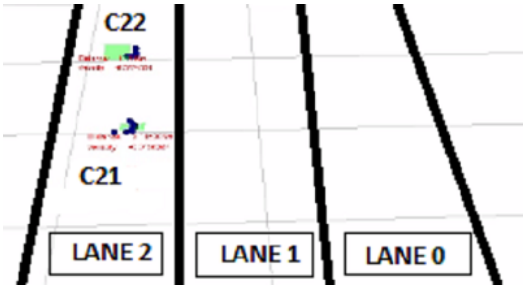
(c) Stage2 (camera).



(d) Stage2 (laser).

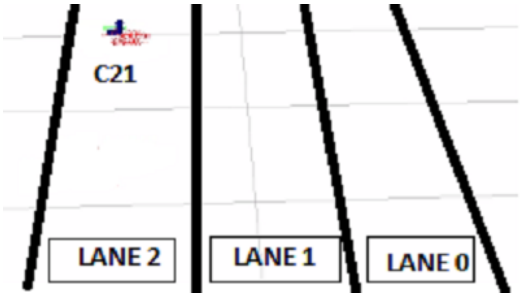


(e) Stage3 (camera).



(f) Stage3 (laser).

(g) Stage4 (camera).



(h) Stage4 (laser).

Fig. 4.22.: Different instances of multiple target-tracking.

4.5 Interactive-multiple-model-based two-stage estimator

4.5.1 Introduction

Using the models proposed in the maneuver tracking algorithm and integrating it with multiple target-tracking, an overall two-stage velocity-estimator based on Interactive-Multiple-Model (IMM) algorithm has been proposed. In this research, the proposed estimator consist of 3 variable-structure models, changing according to the estimated turn-rate parameter. Variable-structure models are models that are not fixed throughout the execution of the algorithm [23]; in our case, the models change with the time-varying turn-rate ω. These state models are linear in terms of the state-vector components, and hence a standard Kalman filter can be implemented for tracking purposes. The performance of the algorithm has been validated using simulations as well as experimentation on a P3-DX mobile robot platform. In this section, an overview of Kalman filtering, IMM filtering along with the proposed two-stage estimator will be discussed.

4.5.2 Kalman filter for velocity estimation

Kalman filtering is an algorithm that operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state [75]. It is optimal in the sense that if the noise is Gaussian, the filter minimizes the mean-squared error of the estimated parameters. The filter finds the best estimate from noisy data that includes filtering out the noise and projecting these measurements onto the state estimate.

The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next noisy error-prone measurement is observed, these estimates are updated using a weighted average. Due to the algorithm's recursive nature, it can run in real-time using only the present input measurements and the previously

calculated state and its associated covariance. The following set of equations describe the filtering algorithm [76]

$$X(k+1) = FX(k) + Gu(k) + E\nu(k) \tag{4.34}$$

$$Z(k) = HX(k) + w(k) \tag{4.35}$$

with parameters as described in Eq. (4.1).

Let us denote the process noise covariance Q and sensor noise covariance R as

$$Q = E[E\nu(k)\nu(k)^T E^T] = E\sigma_\nu^2 E^T \tag{4.36}$$

$$R = E[w(k)w(k)^T] \tag{4.37}$$

Then the solution of Kalman filter is given by the following steps:

$$\textbf{Predicted state:} \quad \hat{X}(k|k-1) = F\hat{X}(k-1|k-1) + Gu(k) \ , \tag{4.38}$$

where $\hat{X}(k|k-1)$ predicts the state $X(k)$ based on the prior information of $X(k-1)$.

$$\textbf{State prediction covariance:} \quad P(k|k-1) = FP(k-1|k-1)F^T + Q \ , \tag{4.39}$$

where the covariance $P(k|k-1)$ associated with the predicted state $\hat{X}(k|k-1)$ is computed.

$$\textbf{Predicted measurement:} \quad \hat{Z}(k|k-1) = H\hat{X}(k|k-1) \ , \tag{4.40}$$

where $\hat{Z}(k|k-1)$ denotes the predicted measurement at k^{th} iteration based on the sensor model and predicted state information.

$$\textbf{Measurement covariance:} \quad S(k) = HP(k|k-1)H^T + R(k) \ , \tag{4.41}$$

where $S(k)$ denotes the covariance associated with the measurement.

$$\textbf{Kalman gain:} \quad K(k) = P(k+1|k)H^T S(k+1)^{-1} \ , \tag{4.42}$$

where $K(k)$ denotes the optimal Kalman gain that determines the significance of the measurements over the predicted state estimate.

$$\textbf{Measurement residual:} \quad v(k) = Z(k) - \hat{Z}(k|k-1) = \tilde{Z}(k|k-1) \ , \tag{4.43}$$

where $v(k)$, the measurement residual indicates the difference between the actual measurement and the predicted measurement.

$$\textbf{Updated state estimate:} \quad \hat{X}(k|k) = \hat{X}(k|k) + K(k)\boldsymbol{\nu}(k) \tag{4.44}$$

$$\textbf{Updated covariance:} \quad P(k|k) = P(k|k-1) - K(k)S(k)K^T(k) \tag{4.45}$$

The updated state estimated $\hat{X}(k|k)$ (in Eq. (4.44)) at each iteration provides an estimation of the state-vector components and the updated covariance $P(k|k)$ (in Eq. (4.45)) provides the covariance associated with the current estimation.

4.5.3 Interactive-multiple-model estimator

"The Interacting-Multiple-Model (IMM) estimator is a suboptimal hybrid filter with one of the most cost-effective hybrid state estimation schemes" [77]. The main feature of this algorithm is its ability to estimate the state of a dynamic system with several behavior modes which can switch from one to another (see Fig :4.23). It is based on the fact that the behavior of a target cannot be characterized at all times by a single model, but a finite number of models can adequately describe its behavior under different scenarios. An IMM estimator has been described as a hybrid filter as it has the following properties of a hybrid system:

1. State that evolves according to a stochastic difference equation model.
2. Model that is governed by a discrete stochastic process, and that switches from one mode to another according to a set of transition probabilities.

The computational requirements of an IMM estimator are nearly linear (in number of models) while its performance is almost the same as that of an algorithm with quadratic complexity. Another advantage is that it can be readily combined with data association techniques [78] to handle outliers [79]. An IMM algorithm can have fixed- or variable-structure state-space models which provides flexibility to characterize a dynamic system. One cycle of the algorithm consists of the following steps [69]

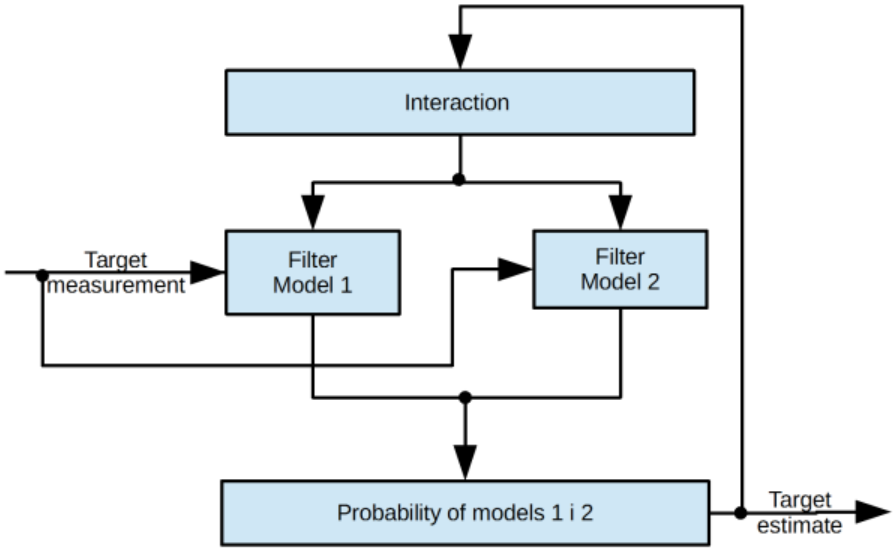


Fig. 4.23.: IMM-based estimator block diagram [80].

1. Probability that mode M_j was in effect at $k-1$, given that mode M_j is in effect at k, conditioned on Z^{k-1} with r number of filters is given as:

$$\mu_{i|j}(k-1|k-1) \triangleq P\{M_i(k-1)|M_j(k), Z^{k-1}\}$$

from which we obtain

$$\mu_{i|j}(k-1|k-1) = \frac{1}{\bar{c}_j} p_{ij}\mu_i(k-1) \ , \tag{4.46}$$

where

$$\begin{aligned} p_{ij} &= P\{M_j(k)|M_i(k-1), Z^{k-1}\} \ , \\ \mu_i(k-1) &= P\{M_i(k-1), Z^{k-1}\} \ , \\ \bar{c_j} &= P\{M_j(k)|Z^{k-1}\} = \sum_{i=1}^{r} p_{ij}\mu_i(k-1) \qquad j = 1, ...r \ , . \end{aligned}$$

and $P\{M_i(k-1)|M_j(k), Z^{k-1}\}$ denotes the probability of mode $M_i(k-1)$ was in effect at $(k-1)^{th}$ iteration given that mode $M_j(k)$ that is in effect at k^{th} iteration with measurement Z^{k-1}.

2. Mixing initial conditions for the filter matched to $Mj(k)$

$$\hat{x}^{0j}(k-1|k-1) = \sum_{i=1}^{r} \hat{x}^{i}(k-1|k-1)\mu_{i|j}(k-1|k-1) \quad j = 1, ...r \tag{4.47}$$

$$\begin{aligned} P^{0j}(k-1|k-1) = \sum_{i=1}^{r} \mu_{i|j}(k-1|k-1)\{P^{i}(k-1|k-1) \\ + [\hat{x}^{j}(k-1|k-1) - \hat{x}^{0j}(k-1|k-1)][\hat{x}^{j}(k-1|k-1) \\ - \hat{x}^{0j}(k-1|k-1)]^{T} \end{aligned} \tag{4.48}$$

where $\hat{x}^{0j}(k-1|k-1)$ and $P^{0j}(k-1|k-1)$ denote the mixed initial state-vector and state covariance respectively. In this stage, the initial estimate $\hat{x}^{0j}$ and its associated covariance P^{0j} is updated with the mode-probability information $\mu_{i|j}(k-1|k-1)$ obtained from the filter output of $(k-1)^{th}$ iteration.

3. Mode matching filtering - Equation 4.49 is the probability of measurement $z(k)$ given the model $M_j(k)$ and previous measurements. The measurement prediction errors and covariance from $z(k)$ can be used to find the likelihood $\Lambda_j(k)$. With $\hat{z}^j$ and S^j as estimate of $z(k)$ and measurement covariance by filter j respectively, we obtain:

$$\Lambda_j(k) = p[z(k)|M_j(k), Z^{k-1}] \ , \tag{4.49}$$

where $p[a|b, c]$ denotes probability of a conditioned on b and c.

4. Mode probability update - Information regarding the current mode probability $\mu_j(k)$ is updated using the following equation

$$\mu_j(k) \triangleq P\{M_j(k)|Z^k\} = \frac{\Lambda_j(k)\bar{c_j}}{\sum_{j=1}^{r} \Lambda_j(k)\bar{c_j}} \quad j = 1, ...r \tag{4.50}$$

This stage involves updating the mode probability $\mu_j(k)$ based on the model likelihoods $M_j(k)$ given the measurements Z^k.

5. State and covariance combination - once the individual model estimates and covariances are computed, they need to be combined using the mode probabilities to obtain the state estimate $\hat{x}(k|k)$ and the state covariance $P(k|k)$ respectively:

$$\hat{x}(k|k) = \sum_{j=1}^{r} \hat{x}^j(k|k)\mu_j(k) \tag{4.51}$$

$$P(k|k) = \sum_{j=1}^{r} \mu_j(k)\{P^j(k|k) + [\hat{x}^j(k|k) - \hat{x}^j(k|k)][\hat{x}^j(k|k) - \hat{x}^j(k|k)]^T \tag{4.52}$$

These stages are performed at an iterative manner to estimate the state and its associated covariance. The model with the highest probability indicates the estimated model (currently in use).

4.5.4 Two-stage estimator for vehicle tracking

A two-stage IMM-based estimator has been proposed to perform multiple target-tracking with application to vehicles in a lane-changing scenario [77]. In this section, we discuss about the integration of multiple target-tracking and curvilinear velocity estimation using a two-stage IMM-based estimator.The block diagram in Fig. 4.24 summarizes the proposed two-stage process.

1. **Stage 1 estimator (turn-rate estimation):** The first stage deals with an adaptive-window-based turn-rate estimation for tracking maneuvering targets (Section 4.3). The estimator can detect abrupt changes in turn-rates within a limited locking delay, and functioning independently, it avoids problem related to non-linear vehicle kinematics, and allows the implementation of standard Kalman filter for tracking.

2. **Model update:** Variable structure models are updated with estimated turn-rates $\{\omega_L, \omega_C, \omega_R\}$ and are utilized in the second stage to perform data association followed by IMM-based velocity estimation.

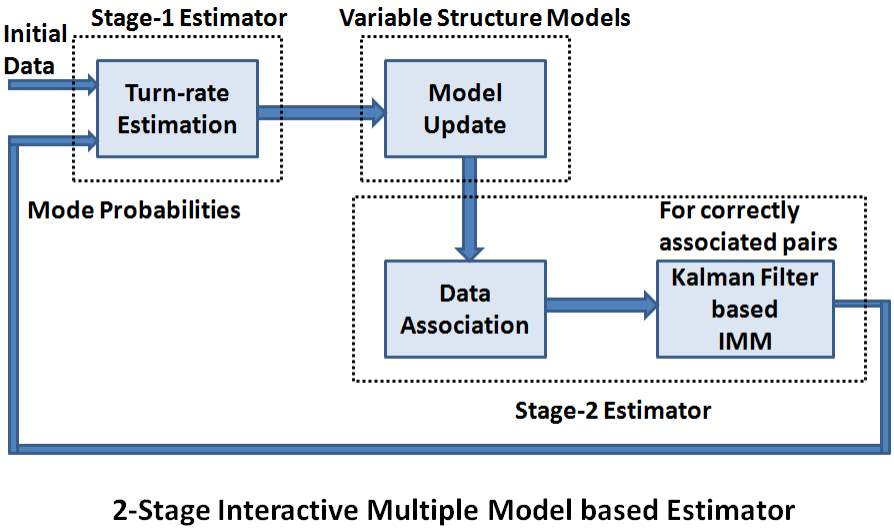


Fig. 4.24.: IMM-based 2-stage velocity-estimator block diagram.

3. **Stage 2 estimator**: This stage of the estimator performs the task of prediction and update of the state parameters based on noisy measurements. It is divided into the following operations:

   (a) **Data association:** As discussed in Section 4.4, data association of the measurements to the target vehicles is an important step for better accuracy and elimination of data loss. Using a modified version of the NN-JPDAF algorithm, each of the measurement is either paired with the appropriate target vehicle or considered as a new target vehicle.

   (b) **Kalman-filter-based IMM velocity estimation:** The valid target-measurement pairs are processed for state prediction using the Kalman-filter-based estimation technique. IMM concepts are utilized in each of these estimation steps for each target vehicles with 5 different models (2 fixed-structure and 3 variable-structure models). Although the governing

equations for all the state models remain the same, the state transition matrices differ based on the turn-rate values

$$\omega = \begin{bmatrix} \omega_{LS} & \omega_L & \omega_C & \omega_R & \omega_{RS} \end{bmatrix} \ . \tag{4.53}$$

At each cycle, the turn-rate is estimated by the probabilistic weighted sum of all the updated turn-rates:

$$\omega_C^{k+1} = \mu_{LS}^k \omega_{LS}^k + \mu_L^k \omega_L^k + \mu_C^k \omega_C^k + \mu_R^k \omega_R^k + \mu_{RS}^k \omega_{RS}^k \ , \tag{4.54}$$

where $\mu = [\mu_{LS}\, \mu_L\, \mu_C\, \mu_R\, \mu_{RS}]^T$ denotes the mode probability matrix of the IMM filter. Once the turn-rate is estimated, the other variable-structure models are updated accordingly. A standard Kalman filter [76] is applied to each model of each pair and an estimate of its position and velocity is obtained.

Overall, the second stage of the estimator can also be referred to as a modified IMM-NNJPDA filter used for velocity estimation of multiple target vehicles on the road. An overall layout of the entire multiple target-tracking algorithm is summarized in Fig. 4.25 that integrates the mathematical details of IMM filter (discussed in Section 4.5.3) along with the maneuvering target-tracking proposed in Section 4.3.

4.5.5 Experimentation with mobile robots

The validation of the proposed algorithm has been carried out using experimentation on mobile robots in simulated lane environments. The sensor used has only 180° field of view and can hence track vehicles that are located in front of the host vehicle. Using ROS visualization tool, the tracking of target vehicles has been well demonstrated.

The first experiment performs tracking with both the host and target vehicles in linear motion. The scenario starts off with the host vehicle tracking four neighboring vehicles and gradually two vehicles move out of the sensor-frame. Vehicles that

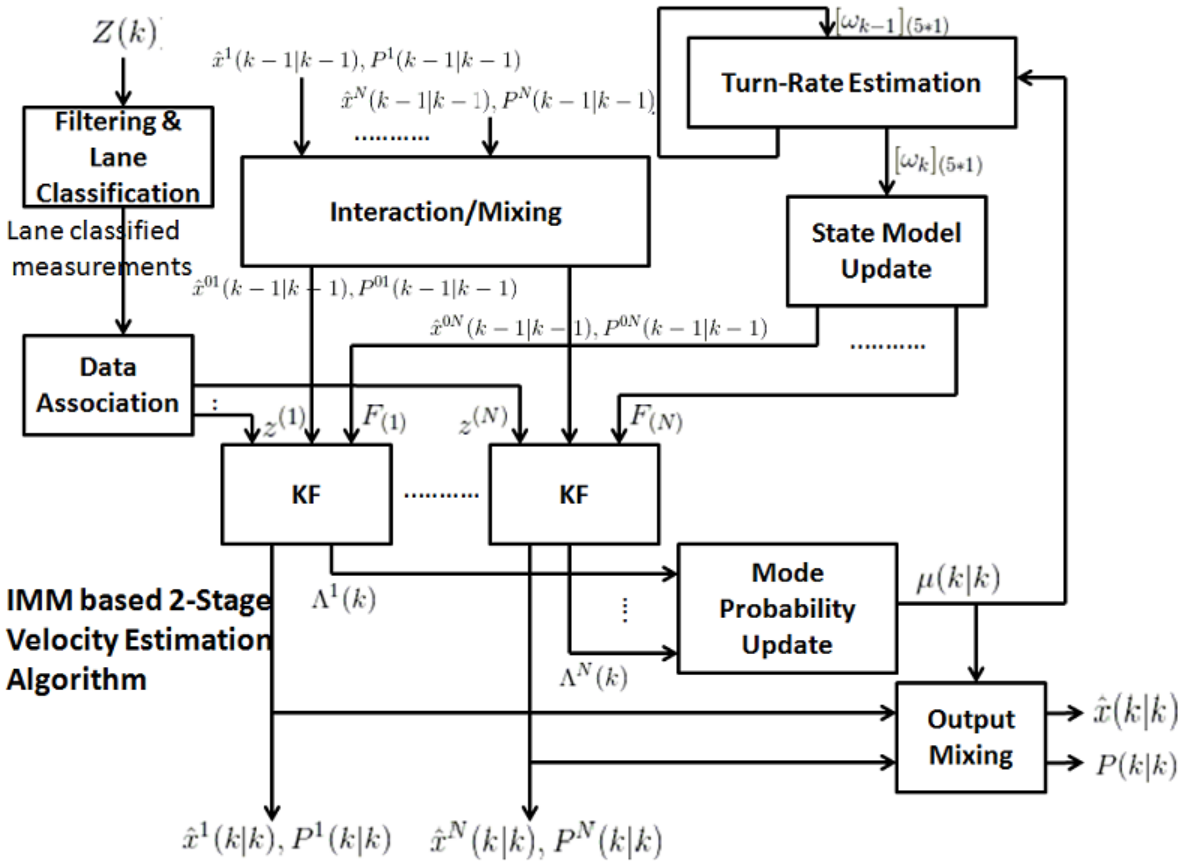
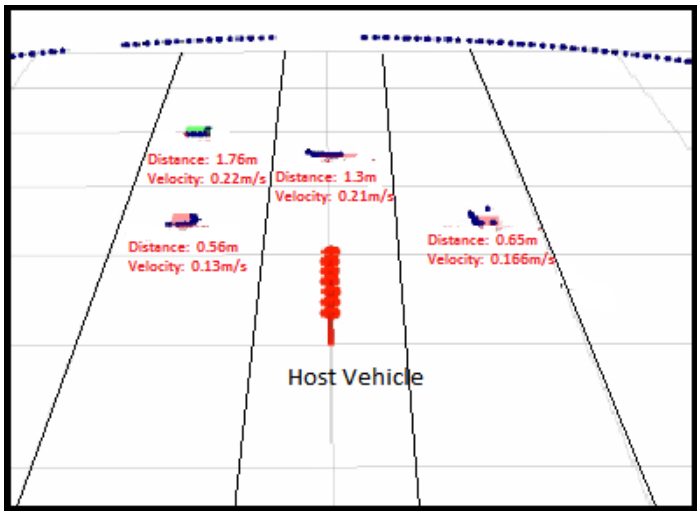


Fig. 4.25.: IMM-based 2-stage velocity-estimator architecture.

play a significant role during lane-changing decisions are tracked as critical vehicles. Figures 5.19(a) and 5.19(c) show visualization of the scenarios with snapshots of the real-world scenes portrayed in Figs. 5.19(b) and 5.19(d).

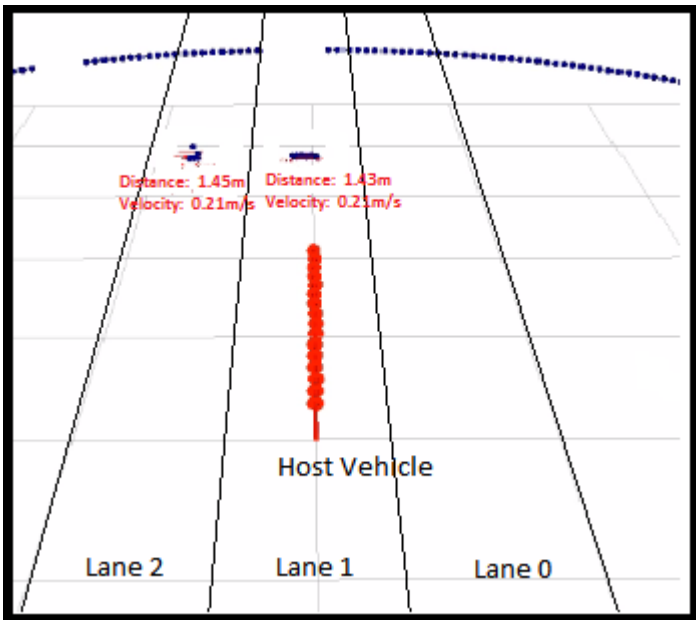
The second experiment performs tracking with the host in linear motion and one of the target vehicles undergoing a lane-change. Figures 4.27(a) and 4.27(c) depict the sensory visualization of the scenarios and the corresponding snapshots of the real-world scenes are observed in Figs. 4.27(b) and 4.27(d).

The third experiment tracks target vehicles in linear motion with the host vehicle undergoing a lane-change. With the sensor in a rotating frame of motion, appropriate

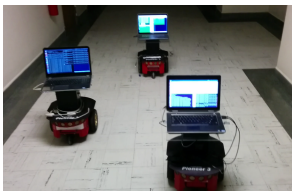


(a) Stage1: ROSViz Visualization

(b) Stage1: Real-world scenario snapshot

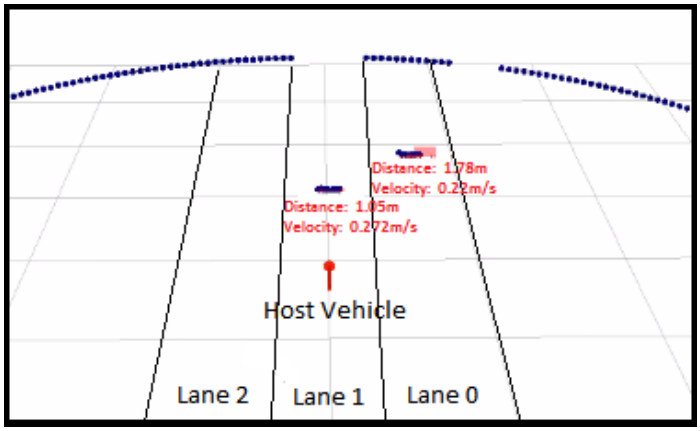


(c) Stage2: ROSViz Visualization



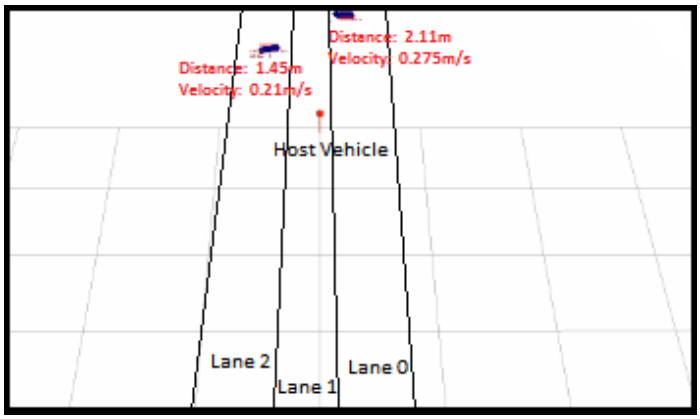
(d) Stage2: Real-world scenario snapshot

Fig. 4.26.: Linear motion for both target and host vehicles.



(a) Stage1: ROSViz Visualization

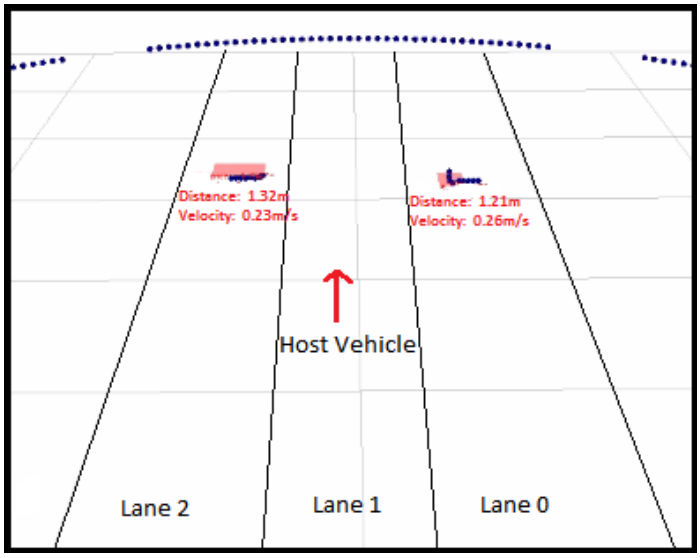
(b) Stage1: Real-world scene



(c) Stage2: ROSViz Visualization

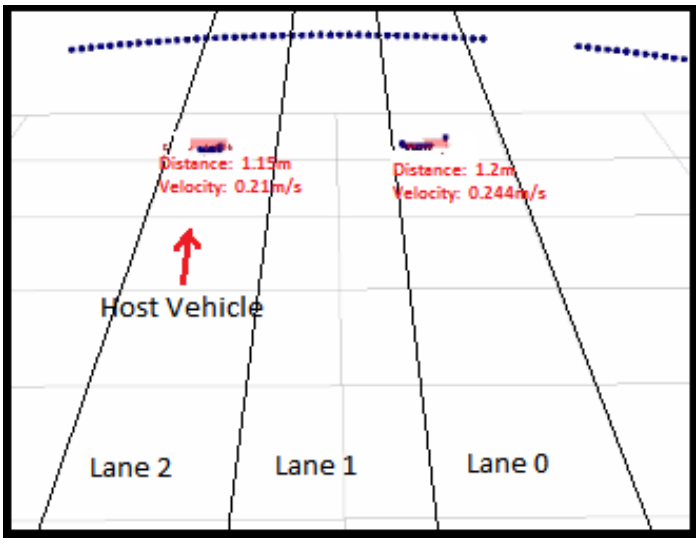
(d) Stage2: Real-world scene

Fig. 4.27.: Target vehicle undergoing curvilinear motion.



(a) Stage1: ROSViz Visualization

(b) Stage1: Real-world scene



(c) Stage2: ROSViz Visualization

(d) Stage2: Real-world scene

Fig. 4.28.: Host vehicle undergoing curvilinear motion.

coordinate transformations need to be performed. Figure 4.28(a) and Fig. 4.28(c) show visualization of the scenario with snapshots of the real-world scene portrayed in Fig. 4.28(b) and Fig. 4.28(d). As observed, the host vehicle is able to track the target vehicles while performing a lane-change maneuver.

4.6 Summary

In this section, a two-stage estimator has been developed for the multiple target-tracking of maneuvering targets catering to the vehicles undergoing linear and curvilinear motion in highway scenarios. The proposed IMM-based algorithm avoids complex non-linear computations and achieves maneuvering target-tracking using the standard Kalman filter. It achieves a root-mean-squared error of 5 cm - 6 cm (position) and 0.25 m/s - 0.3 m/s (velocity) respectively with vehicles simulated to be moving at a speed of 5 m/s - 25 m/s and performing lane-change within 4 s - 7.5 s. The algorithm locks turn-rate up to 10% accuracy within 3s - 4s during abrupt jumps in turn-rate values. The algorithm performs well in tracking the neighboring vehicles during linear and curvilinear maneuvers. The velocity and position information obtained from the tracking algorithm is utilized in determining the risk associated with lane-changing maneuvers and plays a significant role in trajectory planning, details of which are provided in Chapter 5.

5. BEZIER CURVE BASED TRAJECTORY PLANNING FOR LANE CHANGE

5.1 Introduction

Based on the velocity and position information of the target vehicles obtained from the two-stage velocity estimator (Chapter 4), this chapter focuses on the use of Bezier curves to generate trajectories for lane-change maneuvers. The generation of these curvature-continuous trajectories, while satisfying several constraints, have been discussed in this chapter. The velocity and position of the target vehicles help in the assessment of risk associated with the lane-change maneuvers, based on which decisions are taken to perform lane-change under different scenarios. A detailed evaluation of the possible trajectories, optimized to performance-indicators such as travel-time, traffic rules, fuel consumption and comfort has also been provided in this chapter. The task of path-planning needs to take into consideration some environmental factors, few of which are provided by the velocity estimator, and the others are determined based on decision-making algorithms. Details regarding the usage of the information provided by the velocity estimator as well as the design of a decision-making algorithm, have been discussed in the later sections of this chapter. Tracking of target vehicles, followed by trajectory generation for lane-change completes the task of autonomous lane-change maneuver.

5.2 Existing methods in lane-change trajectory generation

5.2.1 Higher-order polynomial based trajectory generation

As observed by Nelson [81], polynomial curves are computationally simple and have closed-form expressions that provide continuous curvatures. For lane-change

maneuvers, Nelson proposed a 5^{th}-order polynomial to describe the lateral position of a vehicle as a function of its longitudinal position.

A total of five constraints (three initial values of position, velocity, and acceleration and two final values of velocity and acceleration) is obtained in the longitudinal direction. In the lateral direction, there are six constraints as both current and target values of position, velocity, and acceleration are specified. These define fourth and fifth-order polynomial curves for the longitudinal and lateral positions, respectively. With the origin of the local coordinate frame at the center of gravity of the vehicle, and X and Y being the longitudinal and lateral axes, respectively, we formulate the equations for the longitudinal and lateral positions ($x(t)$ and $y(t)$) as follows:

$$x(t) = a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \tag{5.1}$$

$$y(t) = b_5t^5 + b_4t^4 + b_3t^3 + b_2t^2 + b_1t + b_0 \ , \tag{5.2}$$

where the coefficients a_0, a_1, ..., a_4, and b_0, b_1, ..., b_5 are calculated using the initial and final constraints. The target values of longitudinal and lateral positions, velocities, and accelerations of each trajectory are described in a predefined parameter table. Although having acceleration constraints to characterize the curve is an advantage, non-parametrization with respect to transition time and ignorance of comfort-ride constraints proves to be disadvantageous. Figure 5.1 shows an example of a trajectory generated using the polynomial curve-based approach.

5.2.2 Trapezoidal acceleration trajectory

While a 5^{th}-order polynomial trajectory has a continuous curvature and a simple closed-form, the required transition time is longer than methods of circular/cosine trajectory approximation [82]. To reduce the transition time and parametrize the lateral jerk explicitly, the trapezoidal acceleration trajectory is designed (see Fig. 5.2)

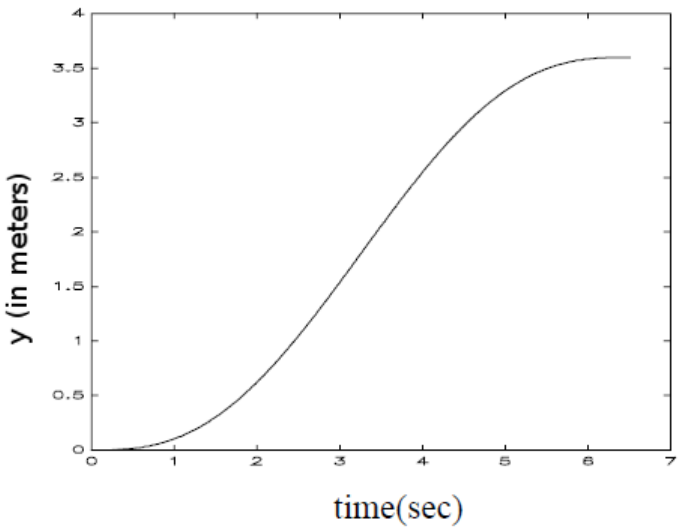


Fig. 5.1.: Polynomial trajectory for lane-change maneuver [81].

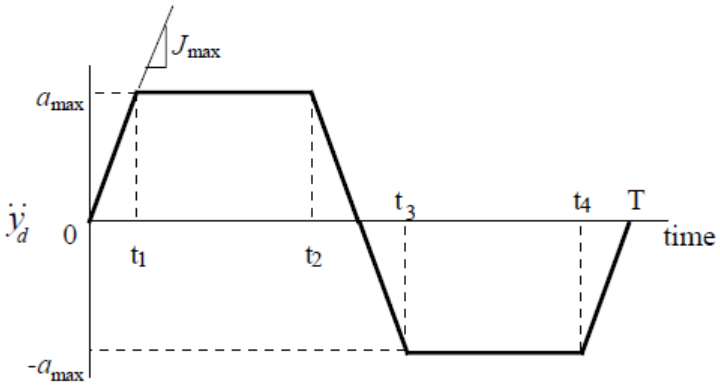


Fig. 5.2.: Trapezoidal acceleration based trajectory for lane-changing [81].

with jerk being the slope of the acceleration profile. The desired lateral acceleration $\ddot{y}_d(t)$ can be written as

$$\ddot{y}_d(t) = J_{max}tu(t) - J_{max}(t-t_1)u(t-t_1) - J_{max}(t-t_2)u(t-t_2) + J_{max}(t-t_3)u(t-t_3) + J_{max}(t-t_4)u(t-t_4) - J_{max}(t-T)u(t-T) \quad (5.3)$$

where J_{max} is the jerk limit and $u(t)$ is a unit step function. With a_{max} as the maximum lateral acceleration and d as the distance between the outer and inner lanes, time parameters of this trajectory, t_1, t_2, t_3, t_4, and T are given as

$$t_1 = \frac{a_{max}}{J_{max}} \ , \quad (5.4)$$

$$t_2 = \frac{-t_1^2 + \sqrt{t_1^4 + 4t_1\frac{d}{J_{max}}}}{2t_1} \ , \quad (5.5)$$

$$t_3 = 2t_1 + t_2 \ , \quad (5.6)$$

$$t_4 = t_1 + 2t_2 \ , \quad (5.7)$$

$$T = 2(t_1 + t_2) \ . \quad (5.8)$$

The resultant trajectory has been known to produce the least possible lateral acceleration on the vehicle. The time required to perform a lane-change and the kinematics of target vehicles can be combined to choose the desired lateral acceleration to meet the constraints for lane-changing on a curved road.

5.3 Bezier curves

5.3.1 Introduction

A Bezier curve is a parametric curve frequently used in fields related to Computer Graphics to model smooth curves. As a Bezier curve is completely contained in the convex hull of its control points, the points can be graphically displayed and used to manipulate the curve intuitively [83]. Affine transformations of the curve, such as translation and rotation, can be produced by applying the respective transform

to the control points of the curve. Quadratic and cubic Bezier curves are the most commononly used, as higher-degree curves are computationally expensive to evaluate. When more complex shapes are needed, low-degree Bezier curves are combined to produce a composite Bezier curve. The curve is drawn with the aid of control points, only two of which fall on the curve. This is quite different from the commonly used interpolation method, where all the points for curve-fitting necessarily lie on the curve.

A Bezier curve of degree n can be represented as

$$P(t) = \sum_{i=0}^{n} B_i^n(t) P_i, \quad t \,\epsilon\, [0,1] \tag{5.9}$$

where P_i denotes the i^{th} control point, with $B_i^n(t)$ being a Bernstein polynomial given by

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i, \quad i \,\epsilon\, \{0,1,...,n\} \ , \tag{5.10}$$

where $\binom{n}{i}$ denotes the combinatorial function, indicating the number of possible ways of selecting i items from n items.

Bezier curves have useful properties that find applications in path-planning:

1. The curve always lies within the convex hull of its control points; that is, $P(t)$ ϵ convex hull $(P_0, P_1, ..., P_n)$.

2. The curve always passes through the start point P_0 and the end point P_n.

3. The starting and ending points on the curve are tangential to the lines joining points $(P_0\ ,\ P_1)$ and $(P_{n-1}\ ,\ P_n)$ at P_0 and P_n, respectively.

Equation (5.9) represents a Bezier curve that starts at $t = 0$ and ends at $t = 1$. We can also denote the Bezier curve defined over an arbitrary parameter interval, $P_{[t_0,t_1]}(t)$, by a modification of Eq. (5.9) as:

$$\begin{aligned} P_{[t_0,t_1]}(t) &= \frac{\sum_{i=0}^{n} \binom{n}{i} (t_1 - t)^{n-i} (t - t_0)^i P_i}{(t_1 - t_0)^n} \\ &= \sum_{i=0}^{n} \binom{n}{i} \left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-i} \left(\frac{t - t_0}{t_1 - t_0}\right)^i P_i \ . \end{aligned} \tag{5.11}$$

If no parameter interval is specified, it is assumed to be $[0, 1]$; that is, $P(t) = P_{[0,1]}(t)$.

In this research, we propose to utilize Bezier curves for performing the task of trajectory generation for lane-change maneuver. However, before providing the details of trajectory generation using Bezier curves, it is necessary to introduce the characteristics and properties of Bezier curves that have been utilized throughout the algorithm.

5.3.2 The de Casteljau algorithm

The de Casteljau algorithm, developed by French mathematician Paul de Casteljau in 1959 [84], has been used to describe a recursive process for dividing a complex curve into simpler segments. The subdivided segments are also Bezier curves. Let $\{P_0^0, P_1^0, ..., P_n^0\}$ denote the control points of $P(t)$. The control points of the segments can be computed by

$$P_i^j = (1-t)P_i^{j-1} + tP_{i+1}^{j-1}, \quad j \,\epsilon\{1, 2, ..., n\} \quad i \,\epsilon\, \{0, 1, ..., n-j\} \tag{5.12}$$

where the time parameter $t \ \epsilon \ (0, 1)$ and $\{P_0^n, P_1^{n-1}, ..., P_n^0\}$ are the control points of one segment and $\{P_0^n, P_1^{n-1}, ..., P_n^0\}$, of the other segment.

An application of the de Casteljau algorithm is that it provides a numerically stable method for the computation of the coordinates of and tangent vector at any point on the curve. When a Bezier curve is continuously subdivided, the collection of control polygons converges to the curve.

5.3.3 Derivatives, continuity and curvature of the Bezier curve

In this section, we shall discuss equations for computing the derivatives, continuity and curvature of Bezier curves. The derivatives of an n^{th}-order Bezier curve, referred

to as the hodograph, can be determined using its control points [85]. For a Bezier curve $P(t) = \sum_{i=0}^{n} B_i^n(t) P_i$, the first derivative can be represented as:

$$\dot{P}(t) = \sum_{i=0}^{n-1} B_i^{n-1}(t) D_i \tag{5.13}$$

where D_i (control points of $\dot{P}(t)$) is given by $D_i = n(P_{i+1} - P_i)$. Geometrically, Eq. (5.13) provides the tangent vector. The first derivative of a Bezier curve $P(t)$ at the end-points can be written as:

$$\dot{P}(0) = n(P_1 - P_0) \tag{5.14}$$

$$\dot{P}(1) = n(P_n - P_{n-1}) \ . \tag{5.15}$$

The higher-order derivatives of a Bezier curve are obtained by using Eq. (5.13) iteratively. The resultant second-order derivatives of $P(t)$ at endpoints are:

$$\ddot{P}(0) = n(n-1)(P_2 - 2P_1 + P_0) \tag{5.16}$$

$$\ddot{P}(1) = n(n-1)(P_n - 2P_{n-1} + P_{n-2}) \ . \tag{5.17}$$

The property of continuity of Bezier curves is very critical, especially in the case of composite Bezier curves, where smaller segments are integrated into a complex path based on the de Casteljau algorithm. Two Bezier curves $P(t)$ and $Q(t)$ are said to be C^k continuous at t_0 if

$$P(t_0) = Q(t_0) \tag{5.18}$$

$$P^{(1)}(1) = Q^{(1)}(1) \tag{5.19}$$

$$\ldots \tag{5.20}$$

$$\ldots \tag{5.21}$$

$$P^{(k)}(t_0) = Q^{(k)}(t_0) \tag{5.22}$$

The curvature of a Bezier curve $P(t) = (x(t), y(t))$ at time t is given by:

$$\kappa(t) = \frac{|\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)|}{(\dot{x}^2(t) + \dot{y}^2(t))^{3/2}} \tag{5.23}$$

where $\kappa(t)$ represents the curvature. The path constructed by two Bezier curve segments, lets say $P_{[t_0,t_1]}(t)$ and $Q_{[t_1,t_2]}(t)$, has continuous curvature for every point on it if $P(t)$ and $Q(t)$ are at least C^2 continuous at t_1. This fundamental property of the Bezier curves has been utilized extensively in the field of path-planning.

5.3.4 Bezier curves for path-planning

Quadratic Bezier curves

The quadratic Bezier curve is one of the most popular types used in path-planning applications. It is defined by three control points c_0, c_1 and c_2 constructed by forming their convex combination. Using time parameter t, we define the two line segments over an interval, say $[0, 1]$,

$$p_{1,1}(t) = (1-t)c_0 + tc_1 \ , \tag{5.24}$$

$$p_{2,1}(t) = (1-t)c_1 + tc_2 \ . \tag{5.25}$$

Forming a convex combination, we obtain

$$p_{2,2}(t) = (1-t)p_{1,1}(t) + tp_{2,1}(t) \ , \tag{5.26}$$

$$p_{2,2}(t) = (1-t^2)c_0 + 2t(1-t)c_1 + t^2c_2 \ , \tag{5.27}$$

where the piecewise linear curve connecting them is called the control polygon of the curve. Two of the control points c_0 and c_2 mark the start and end points, and the third one determines the curvature of the path.

Properties of Bezier curves applicable to lane-changing

1. **Continuity:** If two segments of Bezier curves are C^2 continuous at t_1, then every point on the two Bezier curves is continuous and the two segments can be combined. During lane-changing, the two separate segments, representing the journey till the edge of the current lane, followed by journey to the center of the target lane, can be characterized by two continuous Bezier curves.

2. **Symmetrical Nature:** Bezier curves have the property that if the time parameter t is replaced by $-t$, we obtain a symmetric curve reflected about the origin. This aids in reducing the computation time of a symmetrical path by performing computations for only one half of the total curve.

3. **Tangent Property:** The tangents at the start and end points of a Bezier curve are parallel. During lane-changing, tangents at the curve before and after the maneuver are parallel to each other and to the longitudinal axis, thus indicating that the vehicle moves straight after the completion of lane-change.

Previous work on path-planning using Bezier curves

In recent times, progress has been made in the field of path-planning using Bezier curves [86]. The fact that Bezier curves constructed by large numbers of control points are numerically unstable, together with the piecewise continuity property resulted in approaches that divide the overall path into several smaller paths. In [9], two path-planning algorithms based on Bezier curves are considered with waypoints and corridor constraints. The paper describes the application of useful properties of Bezier curves in the algorithms to generate the reference trajectory for vehicles, while satisfying user-defined path constraints.

In [10], a piecewise Bezier path was manipulated by control points of the bounding polygon. Decisions regarding lane-change are based on the calculation of a minimum safe lane-change distance, followed by path-planning, satisfying the algorithm-imposed curvature and state constraints and also meeting the turn-rate requirement. However, this work assumes a constant time for lane-change, which is not applicable to all lane-changing scenarios.

5.4 Proposed method for Bezier-cruve-based trajectory generation

This section provides details of the proposed method for trajectory generation. Different concepts utilized in the algorithm are discussed, followed by the design of the algorithm.

5.4.1 Maximum heading angle

The heading angle (ϕ) is the angle between the direction of motion and the longitudinal axis. While performing a lane-change, we need to set an upper bound (ϕ_{max}) to ensure safe and comfortable motion without introducing a lot of yaw into the system. From Fig. 5.3, we can obtain the relation:

$$\tan\phi = \frac{b}{a-c} \tag{5.28}$$

where a and b denote half the width of a lane and half the distance covered during lane-change, respectively. Approximating S as the length of the path and $2a$ as the longitudinal distance, we can formulate a lower bound for the minimum path of lane-change maneuver as:

$$S \leq \frac{2b}{\tan\phi_{max}(1-p)} \ , \tag{5.29}$$

where S denotes the length of the planned path, and p the fraction of $(s/2)$ that forms the third control point. Considering $b = 1.85\,m$, $\phi_{max} = 35^\circ$ and $p = 0.425$, we obtain $9.1\,m$ as the minimum lane-changing distance. However, ϕ_{max} is a user-defined parameter and can be changed according to the user requirement.

5.4.2 Length of the path

Although the initial approximation to the length of the trajectory generated is based on linear motion, it is important to calculate the actual length of the path. The following approach is made towards computing the length of the curve based on the given control points p_0, p_1, and p_2 with coordinates $(p_{0x}, p_{0y}), (p_{1x}, p_{1y}), (p_{2x},$ and $p_{2y})$, respectively:

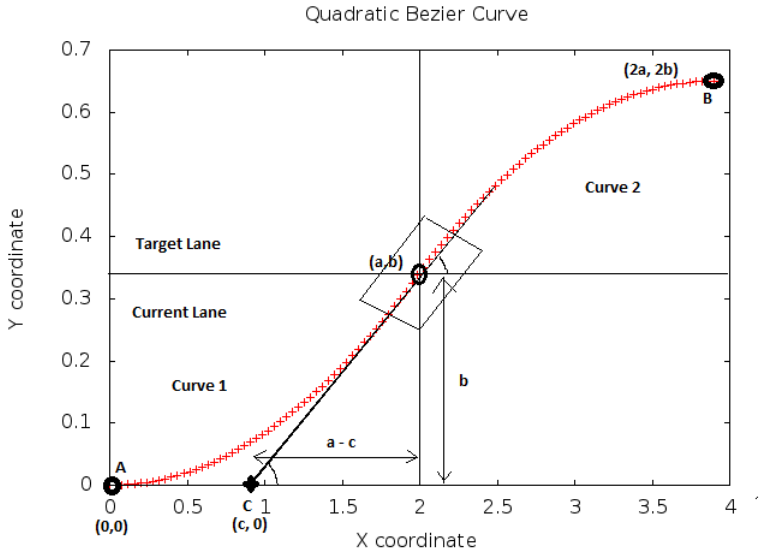


Fig. 5.3.: Maximum heading angle of vehicle.

$$a_x = p_{0x} - 2p_{1x} + p_{2x} \ , \tag{5.30}$$

$$a_y = p_{0y} - 2p_{1y} + p_{2y} \ , \tag{5.31}$$

$$b_x = 2p_{1x} - 2p_{0x} \ , \tag{5.32}$$

$$b_y = 2p_{1y} - 2p_{0y} \ , \tag{5.33}$$

$$A = 4(a_x^2 + a_y^2) \ , \tag{5.34}$$

$$B = 4(a_x b_x + a_y b_y) \ , \tag{5.35}$$

$$C = b_x^2 + b_y^2 \ , \tag{5.36}$$

$$L = \int_0^1 \sqrt{At^2 + Bt + C}dt \tag{5.37}$$

Using the above integral, we obtain:

$$L = \frac{1}{8A^{3/2}} 4A^{3/2}\sqrt{A+B+C} + \frac{1}{8A^{3/2}} 2\sqrt{A}B(\sqrt{A+B+C} - \sqrt{C}) + \frac{1}{8A^{3/2}}(4CA - B^2)\log\left(\frac{2\sqrt{A} + (B/\sqrt{A}) + 2\sqrt{A+B+C}}{(B/\sqrt{A}) + 2\sqrt{C}}\right) \quad . \tag{5.38}$$

where L denotes the length of the path covered during a lane-change maneuver. Once the length of the curve can be computed, accuracy of the risk associated with lane-change maneuvers can be improved. Thus, length of Bezier curve for the generated trajectory helps in the decision-making task of determining the feasibility of lane-change maneuvers .

5.4.3 Re-parametrization of curve

One important aspect of the Bezier curve that needs to be considered is that, for equal intervals in time, equal spacing of points on the curves is not obtained; that is, if the curve is divided into n parts at Δt interval, then the length ΔS obtained changes at every time interval. As shown in Fig. 5.4, the points appear to be more closely-spaced in the initial phase (near $t = 0$) and gradually spread out (as $t \to 1$).

For the purpose of re-parametrization, we consider small elements of the curve and assume the curve between them is approximately linear. Once the elements based on equal intervals of time are created, the value of parameter t can be computed by linear interpolation using the parameter table corresponding to the spacing based on equal intervals of time (see Fig. 5.5). The smaller the time interval of the curve, the higher the interpolation accuracy obtained for the curve. Equation (5.39) shows the steps to perform linear interpolation.

$$\frac{d_j - d_{j-1}}{\Delta t} = \frac{r - d_{j-1}}{t - t_{j-1}} \ ,$$

from which t can be computed as

$$t = \Delta t * \left(\frac{r - d_{j-1}}{d_j - d_{j-1}}\right) + t_{j-1} \ , \tag{5.39}$$

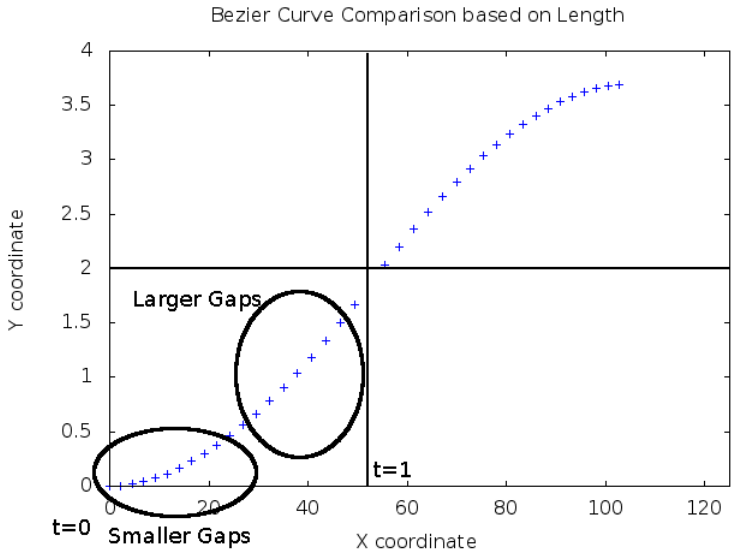


Fig. 5.4.: Need for re-parametrization of curve.

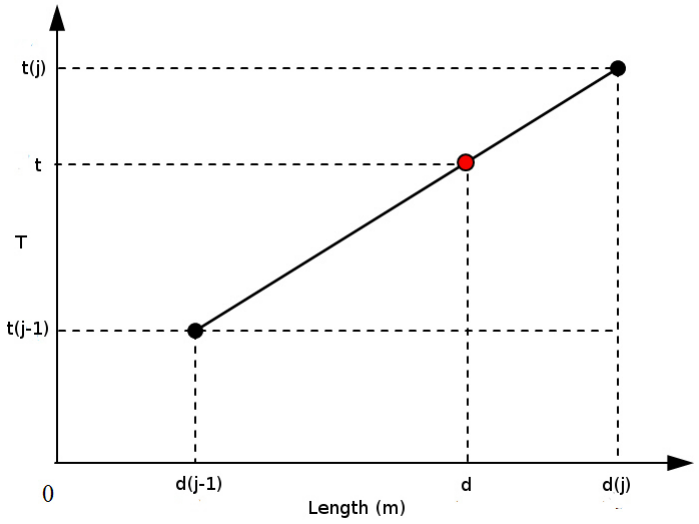


Fig. 5.5.: Linear interpolation.

where r is the required length, d_j and d_{j-1} are the lengths in the neighborhood of the required length and t_j and t_{j-1} are the time parameter values corresponding to d_j and d_{j-1}, respectively.

5.4.4 Relationship among control point, velocity and time for lane-change

For this research, a piecewise quadratic Bezier curve will be considered for generating a path for lane-changing. This implies that we need to locate three control points to generate the trajectory. The start and end points are used as two of the control points. However, it is the third control point that determines the curvature of the path and the angular turn-rate of the vehicle. An extensive comparison of different ranges of time and control point values was performed using a recommended time range of 3 s-8 s [87].

With p as a fraction of half the distance taken for lane-change used for the third control point, we compute different possibilities of p for different velocities (see Table 5.2) and different time intervals (see Table 5.1). Based on a comparison performed on simulated data, we selected a time window of 4.5 s - 7.5 s and a p value of 0.425 to be used for trajectory generation.

Table 5.1: Control points based on velocity for a given time range.

Velocity (m/s^{-1})	**p**
5	0.425 - 0.45
10	0.425 - 0.5
15	0.425 - 0.475
20	0.425 - 0.45
25	0.425 - 0.45

Table 5.2: Control points based on time for a given velocity range.

Time (s)	**p**
4.5	0.425 - 0.45
5	0.425 - 0.45
5.5	0.425 - 0.45
6	0.425 - 0.475
6.5	0.425 - 0.45
7	0.425 - 0.475
7.5	0.425 - 0.45

5.4.5 Risk assessment

In order to detect the possibility of a collision, analysis of the traffic conditions around the vehicle needs to be done. Traffic indicators easily understandable by the driver are characterized using a time value to describe the situation. Two main types of danger exist when considering a lane with traffic: 1) low inter-vehicle distance and 2) high relative velocity. The traffic indicators show that the first type is easily detected by the inter-vehicular time (TIV), while the second can be estimated using the time to collision (TTC).

1. Time To Collsion (TTC): Hayward [88] defines the TTC as "the time required for two vehicles to collide if they continue at their present velocity and on the same trajectory". The TTC formula is:

$$TTC = \frac{D}{V_{lead} - V_{lag}} \ , \tag{5.40}$$

where V_{lead} and V_{lag} denote the velocities of the lead and lag vehicles and D denotes the gap between them. This criterion needs to be utilized only when the

lag velocity exceeds the lead velocity. A threshold of 2 s, used in most existing collision-mitigation systems is being considered for the proposed approach.

2. Inter-vehicular Distance: The TTC itself is not sufficient to describe the risk associated with the situation; for instance, when two vehicles are close to each other with the same velocity, the situation can be dangerous irrespective of a large TTC value. In order to take into account such situations, safety is enhanced by introducing inter-vehicle time as:

$$TIV = \frac{D}{V_{lag}} \ . \tag{5.41}$$

This parameter is often used in regulations (2 s rule used in many of the states in USA).

5.4.6 Modes for lane-change

Figure 5.6 indicates some of the conventional spacing available in the driving scenarios. Most of these factors generated by the two-stage velocity estimator (Chapter 4) need to be considered:

1. Distance between the vehicles in the target lane (availability of a gap).
2. Velocities of the lead and lag target vehicles in the target lane.
3. Velocities of the vehicles in the current lane (front and maybe rear).
4. Distance to the target vehicles in the current lane.
5. Time taken for the lane-change maneuver.
6. Distance covered by the host vehicle during the lane-change.

Most of the decision-making on feasibility of lane-change is dependent on the velocities of the target vehicles in the current and target lanes as well as the availability of a gap. Although the target vehicles are moving with uniform velocity, it is possible

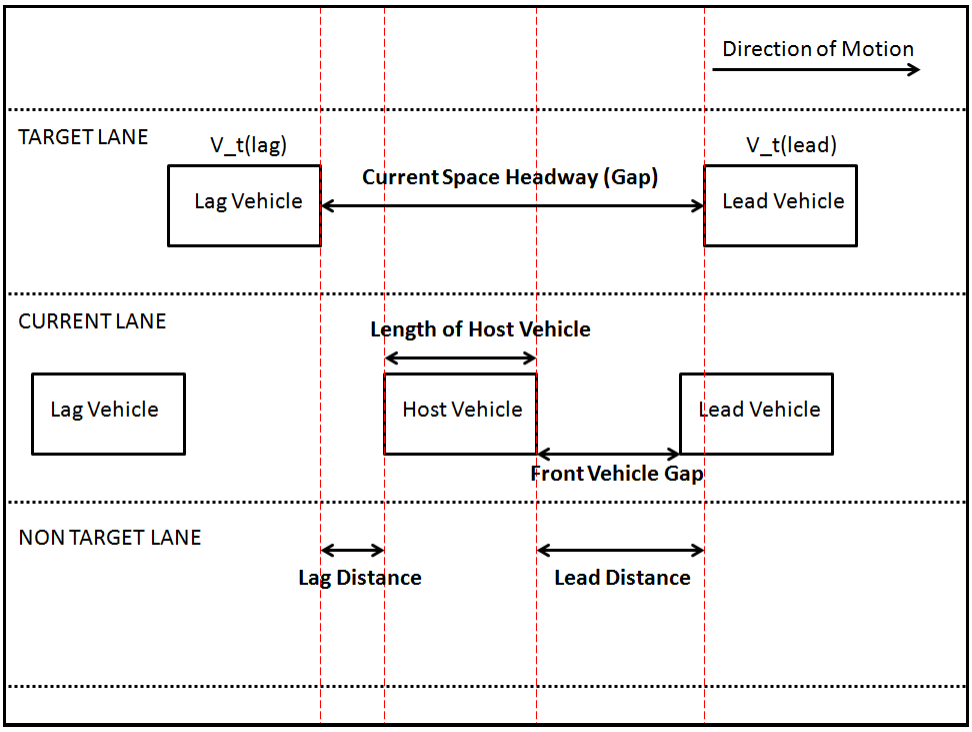


Fig. 5.6.: Lane-gap considerations.

that the gap between them either decreases or increases based on the relative velocity. In this research, two modes have been considered for performing the lane-change:

1. Uniform velocity mode: The host vehicle undertakes the entire lane-change maneuver at a uniform velocity (the same velocity at which it was moving in the current lane).

2. Accelerated mode: The host vehicle accelerates/decelerates during the lane-change maneuver to accommodate for variations in the gap between the target vehicles.

Lane-change is considered to be feasible if either of the modes ensures a safe lane-change. The developed algorithm can either work as an advisory algorithm [89];

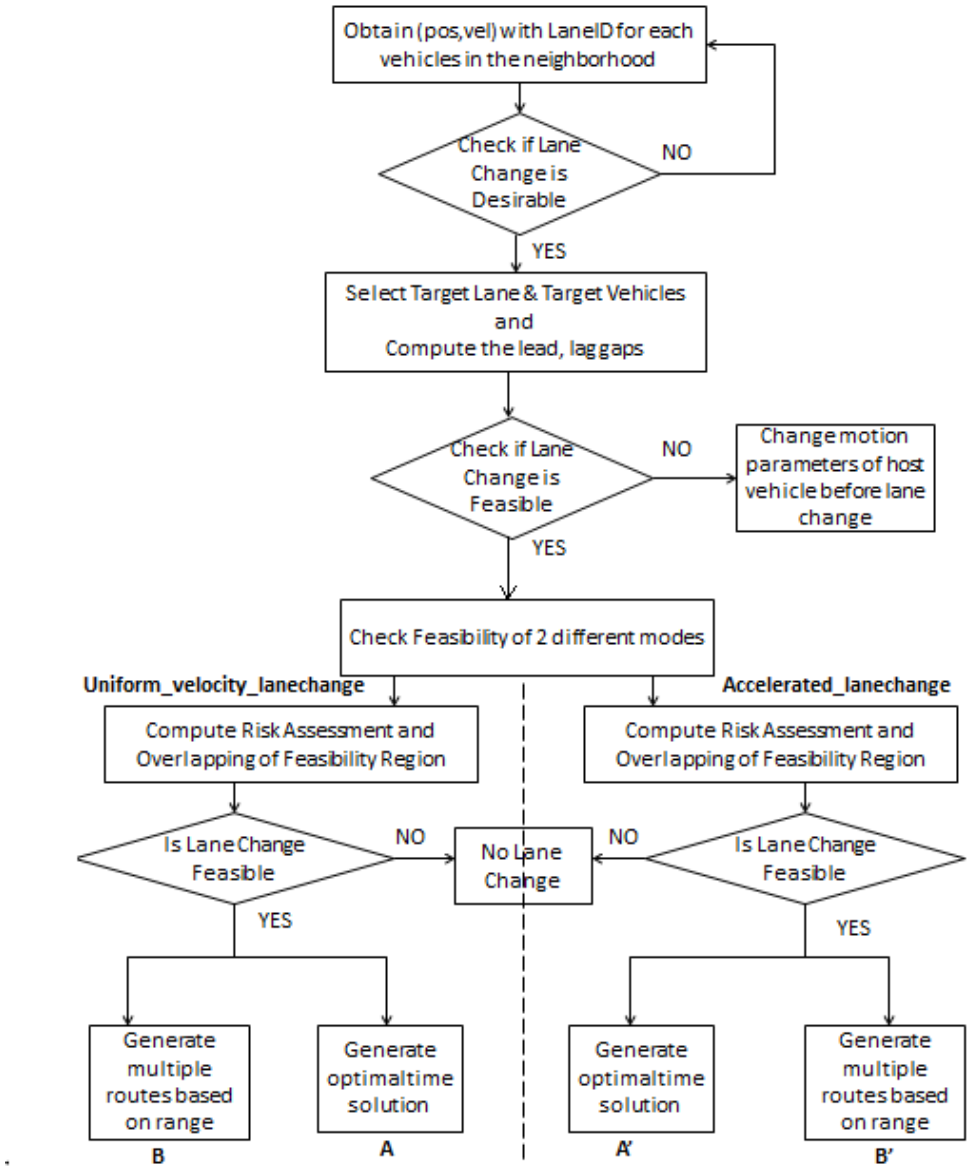


Fig. 5.7.: Lane-change decision-making flowchart.

that is, it can generate multiple paths and allow the driver to select one, or it can generate a solution based on maximization of time within the feasible limit [90]. A mode-independent check is performed to ensure the feasibility of lane-change:

$$V_H t - \frac{1}{2} a t^2 \leq d_{lead} + d_{lag} \ , \tag{5.42}$$

where t is the time taken for lane-change, V_h is the velocity of the host vehicle, d_{lead} and d_{lag} are the distances to the lead and lag target vehicles, respectively. If the gap between the target vehicles is diverging then the feasibility is checked at maximum possible time. If the gap is converging, feasibility should be checked for the least possible time.

5.4.7 Uniform velocity mode of lane-change maneuver

In the uniform velocity mode, the host vehicle undertakes a lane-change at the same velocity at which it was moving in the current lane. The algorithm aims to obtain the optimal time that would minimize the risk associated with the lane-change (TIV and TTC). A simple solution to this would be to maximize the time for lane-change within the feasible limit obtained from the risk assessment parameters. TTC and TIV can be formulated using the velocities of the target vehicles and the gap between them as:

$$TIV = \frac{d + (v_f - v_b)t}{v_f} = \frac{d}{v_f} + \frac{v_f - v_b}{v_f} t \ . \tag{5.43}$$

$$TTC = \frac{d - (v_b - v_f)t}{(v_b - v_f)} = \frac{d}{(v_b - v_f)} - t \ , \tag{5.44}$$

where v_f and v_b indicate the velocities of the front and rear vehicles, respectively, d is the distance between the two vehicles before lane-change and t is the time taken for the lane-change maneuver. The objective of the algorithm is to maximize the time for lane-change.

$$\begin{aligned} \underset{t}{\text{maximize}} \quad & (TIV + TTC) \\ \text{subject to} \quad & T_1^{min} \leq t \leq T_1^{max} \\ & T_2^{min} \leq t \leq T_2^{max} \ , \end{aligned} \tag{5.45}$$

where T_1^{min} and T_1^{max} are given as $4.5s$ and $7.5s$ respectively. The range $[T_2^{min}, T_2^{max}]$ is based on the risk assessment of the surroundings during lane-change maneuver (TTC and TIV).

A detailed analysis of the TIV factor can be performed as:

$$TIV = \frac{d + (v_f - v_b)t}{v_f} \geq threshold_{TIV} \ . \tag{5.46}$$

As mentioned previously, we consider the threshold for TIV to be $1s$.

$$\frac{d + (v_f - v_b)t}{v_f} \geq 1 \ ,$$

from which we obtain,

$$t \geq \frac{v_f - d}{v_f - v_b} \quad , \quad v_f > v_b \qquad or \qquad t \leq \frac{v_f - d}{v_f - v_b} \quad , \quad v_f < v_b \ . \tag{5.47}$$

In case the two velocities v_f and v_b are equal, then the condition $d_{lag} \geq v_f$ will always ensure that TIV is satisfied.

A detailed analysis of the TTC factor can be performed as:

$$TTC = \frac{d - (v_b - v_f)t}{(v_b - v_f)} \geq threshold_{TTC} \ . \tag{5.48}$$

As mentioned previously, we consider the threshold for TTC to be $2s$.

$$\frac{d - (v_b - v_f)t}{(v_b - v_f)} \geq 2$$

from which we obtain,

$$t \leq \frac{d}{v_b - v_f} - 2 \quad , \quad v_f < v_b \ . \tag{5.49}$$

TTC computations are required only when the vehicle behind is moving at a faster velocity than the vehicle in front ($v_f < v_b$). TTC is responsible for the upper bound

of the time while TIV can have an impact on both the upper and lower bounds of the timing considerations for lane-change.

Combining Eq. (5.43) and Eq. (5.44), we obtain

$$\begin{aligned} TTC + TIV &= \frac{d}{v_f} + \frac{v_f - v_b}{v_f} t + \frac{d}{(v_b - v_f)} - t \\ &= K_1 + K_2 t \end{aligned} \tag{5.50}$$

where

$$K_1 = \frac{d}{v_f} + \frac{d}{(v_b - v_f)} \ , \qquad K_2 = \frac{v_f - v_b}{v_f} - 1 \ .$$

Ignoring the constant term, we generate a simple maximization problem as:

$$\begin{aligned} &\text{maximize} && Kt \\ &\text{subject to} && T_1^{min} \leq t \leq T_1^{max} \\ &&& T_2^{min} \leq t \leq T_2^{max} \end{aligned} \tag{5.51}$$

with the value of K being either positive or negative depending on the velocities of the lead and lag vehicles. Once the time for lane-change and the velocity of the host vehicle during lane-change are known, along with the control points of the Bezier curve, it becomes fairly simple to generate the trajectory and initiate the motion commands for undertaking the lane-change.

Figure 5.8 shows the flow of the uniform velocity lane-change maneuver algorithm. It results in two forms of outputs: one that generates multiple paths and provides advisory input to the driver and the other that maximizes time and provides a single trajectory with the associated motion commands. The $sgn(.)$ function used in the algorithm denotes the value of 1 or -1 based on the sign of the parameter.

5.4.8 Accelerated/decelerated mode of lane-change maneuver

There is a possibility that a lane-changing scenario may require an accelerated or decelerated lane-change. It is particularly applicable when the relative arrangement of the vehicles as compared to the total available gap is not sufficient.

Algorithm 4 Uniform velocity lane-change.

Initialize $T_1^{min}, T_1^{max}, T_2^{min}, T_2^{min}, t_i^{min} = T_1^{min}, t_k^{max} = T_1^{max} \quad i = 1, 2; k = 1, 2, 3, 4$

********** TTC risk assessment starts **********

if $(V_{T1} < V_H)$

$t_1^{max} = \frac{d_{lead} - 2(V_H - V_{T1})}{(V_H - V_{T1})}$

if $(V_{T2} > V_H)$

$t_2^{max} = \frac{d_{lag} - 2(V_{T2} - V_H)}{(V_{T2} - V_H)}$

********** TTC risk assessment ends **********

********** TIV risk assessment starts **********

$TIV_{lead} = \frac{V_H - d_{lead}}{V_{T1} - V_H}; \qquad TIV_{lag} = \frac{V_{T2} - d_{lag}}{V_H - V_{T2}}$

Considering $sgn(V_{T1} - V_H)$, assign $\{t_1^{min}, t_3^{max}\}$

Considering $sgn(V_H - V_{T2})$, assign $\{t_2^{min}, t_4^{max}\}$

********** TIV risk assessment ends **********

$T_2^{min} = max\ (t_i^{min}) \quad i = 1, 2; \qquad T_2^{max} = min\ (t_k^{max}) \quad k = 1, 2, 3, 4$

if $((T_2^{min} > T_2^{max})\ ||\ (T_2^{min} > T_1^{max})\ ||\ (T_2^{max} < T_1^{min})$

lane-change not feasible

else

$T_{min} = max\ (T_1^{min}, T_2^{min}); \qquad T_{max} = min\ (T_1^{max}, T_2^{max})$

Generate advisory paths using the time limits (T_{min}, T_{max})

Compute $K = TTC_{lead} + TTC_{lag} + TIV_{lead} + TIV_{lag}$

Compute $T_{selected}$ by solving the time maximization problem

Generate trajectory with time and velocity of host vehicle $(T_{selected}, V_H)$

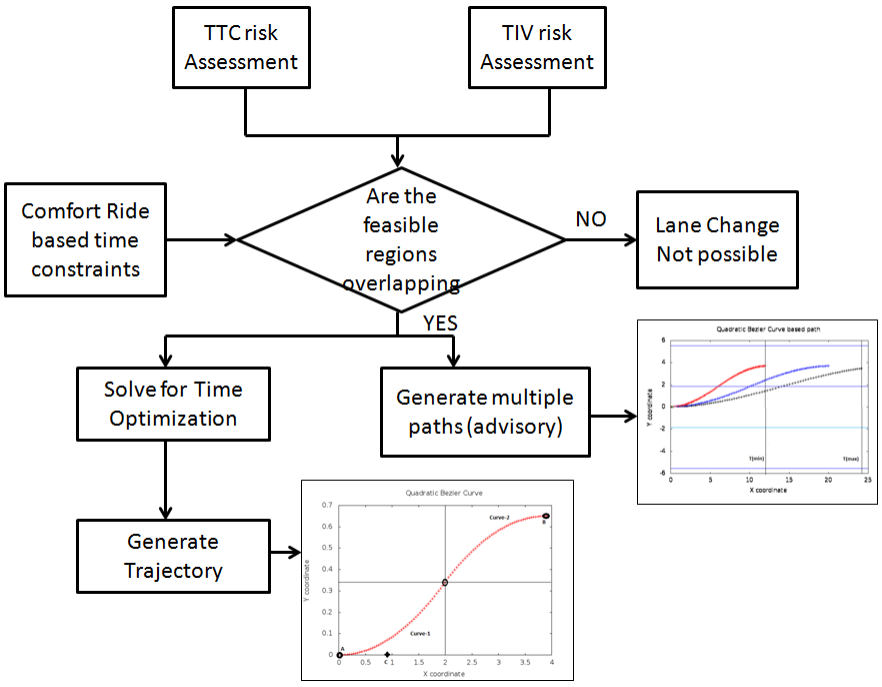


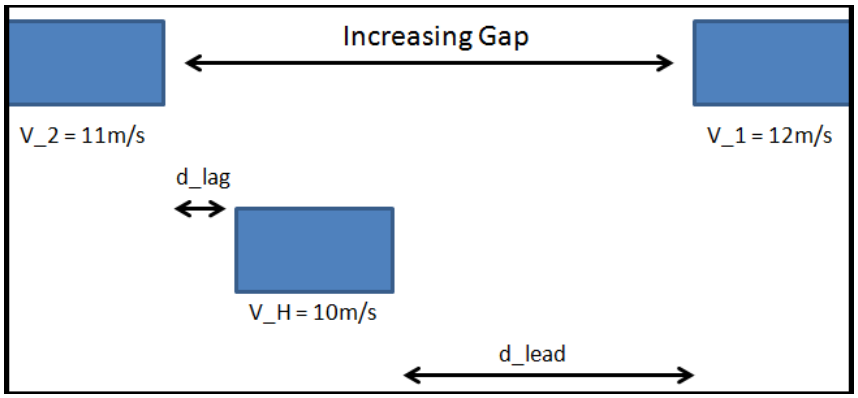
Fig. 5.8.: Uniform velocity mode lane-change.

In Fig. 5.9(a), the host vehicle is very close to the lag vehicle which requires acceleration during lane-change to increase the lag-distance. Hence trajectory needs to be generated in accelerated mode. On the other hand, Fig. 5.9(b) shows that the host vehicle needs to decelerate during lane-change to increase the lead distance.

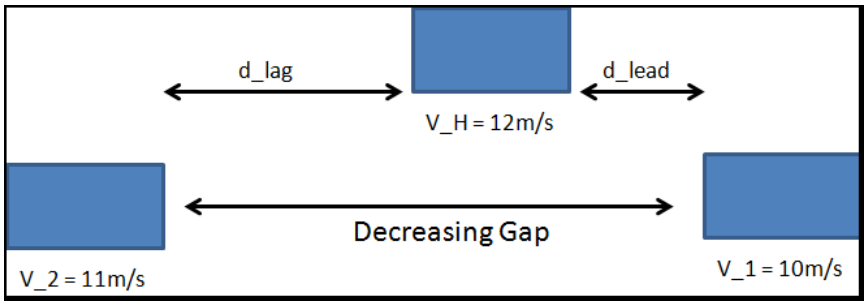
Effect of acceleration/deceleration on turn-rate

An important aspect of this research is to take into account the passenger's comfort-level, which is dependent on the turn-rate of the host vehicle. Hence, it becomes necessary to determine the variation in turn-rate with modes of lane-change. During an accelerated lane-change,

$$S = V_H t + 0.5at^2 \ , \tag{5.52}$$



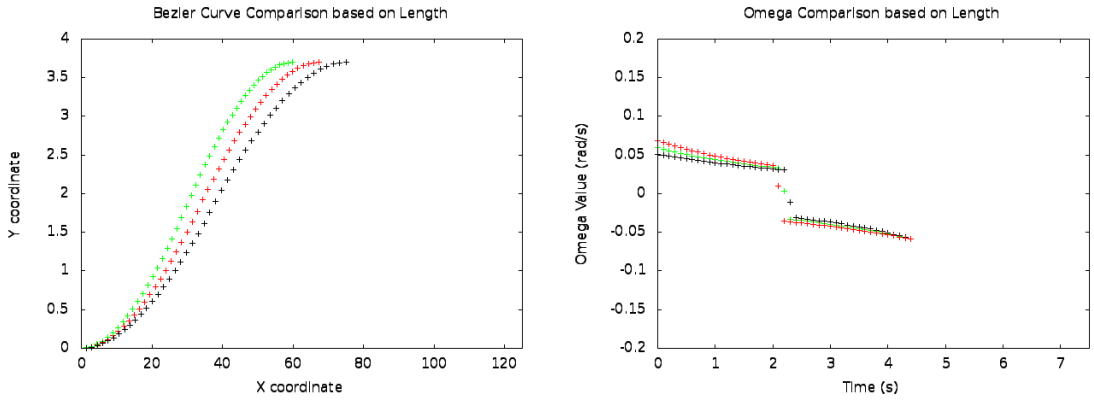
(a) Host close to lag vehicle.



(b) Host close to lead vehicle.

Fig. 5.9.: Requirement for accelerated/decelerated lane-change.

where S can be approximated as the length of the path, a is the acceleration produced during time t. During an accelerated lane-change, the turn-rate value reduces. During deceleration, the length of path covered is less, and hence the turn-rate would have a tendency to increase as compared to the uniform velocity case. Figure 5.10(a) shows the different trajectories generated with different values of acceleration. Figure 5.10(b) shows the variation of turn-rate for each of the associated trajectories. For deceleration cases, it is desirable to keep the deceleration to lower values for a longer duration of lane-change time to ensure better comfort-ride.



(a) Different trajectories in acceleration mode. (b) Omega variations in accelerated mode.

Fig. 5.10.: Accelerated mode trajectories and associated turn-rates.

Problem formulation for accelerated mode lane-change

Applying the velocities of the target and host vehicles, the following conditions are generated during risk-assessment to satisfy the TTC and TIV criteria:

$$\frac{d_{lag} + V_H t + 0.5a * t^2 - V_{lag} t}{V_{lag}} \geq 1 \ , \tag{5.53}$$

$$\frac{d_{lag} + V_H t + 0.5a * t^2 - V_{lag} t}{V_{lag} - V_H - at} \geq 2 \ , \tag{5.54}$$

$$\frac{d_{lead} + V_{lead} t - V_H t - 0.5a * t^2}{V_H + at} \geq 1 \ , \tag{5.55}$$

$$\frac{d_{lead} + V_{lead} t - V_H t - 0.5a * t^2}{V_H + at - V_{lead}} \geq 2 \ , \tag{5.56}$$

where V_H, V_{lead} and V_{lag} are the velocities of the host, lead, and lag target vehicles respectively, d_{lag} and d_{lead} are the lag and lead distance to the host vehicle, a is the acceleration/deceleration involved, and t is the time for the lane-change.

It should be noted that change in velocity of the host vehicle should not cause inconvenience to the target vehicles. We consider a bound on the velocity of the host vehicle at the completion of the lane-change based on the maximum deceleration and TIV considerations:

$$V_{lag} - \hat{a}_{max} t_{TIV} - at \leq V_H \leq V_{lead} + \hat{a}_{max} t_{TIV} - at \tag{5.57}$$

where $\hat{a}_{max}$ denotes the magnitude of the maximum deceleration and t_{TIV} indicates the time parameter based on TIV (in our case, $TIV = 1s$).

Time and acceleration parameters are bounded within the following limits:

$$4.5 \leq t \leq 7.5\ s \tag{5.58}$$

$$-1.5 \leq a \leq 1.5\ m/s^2\ . \tag{5.59}$$

The objective is to obtain a range of values for time t and acceleration a that would satisfy these constraints. Once the range is obtained, the advisory mode can generate multiple paths and the co-pilot mode selects a good (a, t) combination for the given scenario. This problem can also be formulated as a non-linear optimization problem with non-linear constraints:

$$\begin{aligned} &\text{minimize} && K \\ &\text{subject to} && C(x) \leq 0 \\ & && LB \leq x \leq UB \end{aligned} \tag{5.60}$$

where $C(x)$ is a function comprising all the non-linear inequality constraints with $x = [a \quad t]^T$, LB and UB defining the bounds of x and K being any constant (as we need only the feasible solution space and not a particular value of x). Essentially, we are trying to find the solution space that would provide the range of values for acceleration and time parameters of the lane-change. Solving for optimization of a constant, we would obtain a value of x, closest to the initial guess within the solution space.

The method mentioned above for finding the x (solution space of acceleration and time) for the given constraints might have implementation issues; that is, it would

be difficult to come up with optimization algorithms that can solve it in real time. If a closed-form solution or a non-iterative solution is found, the optimization problem can be tackled with great efficiency. An alternative approach to the optimization problem is through reducing the dimensionality of the solution space; that is, solving for the acceleration parameter for a fixed time, $x = [a]$. The problem is thus reduced to solving a system of inequalities for a single variable.

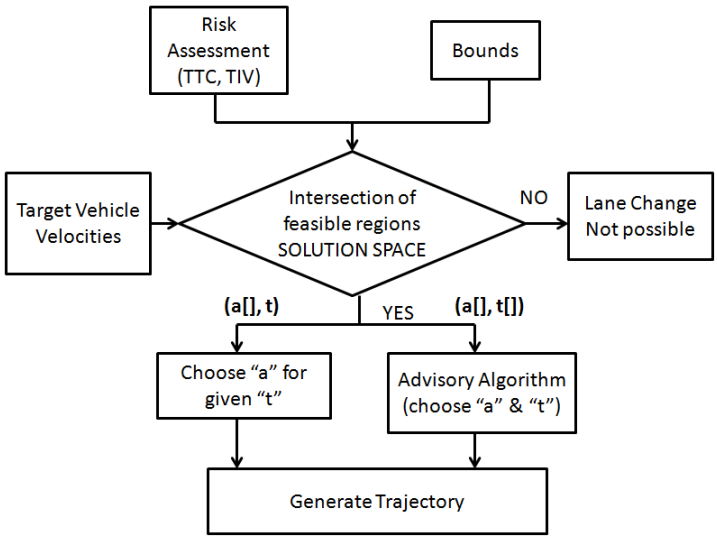


Fig. 5.11.: Accelerated lane-change flowchart.

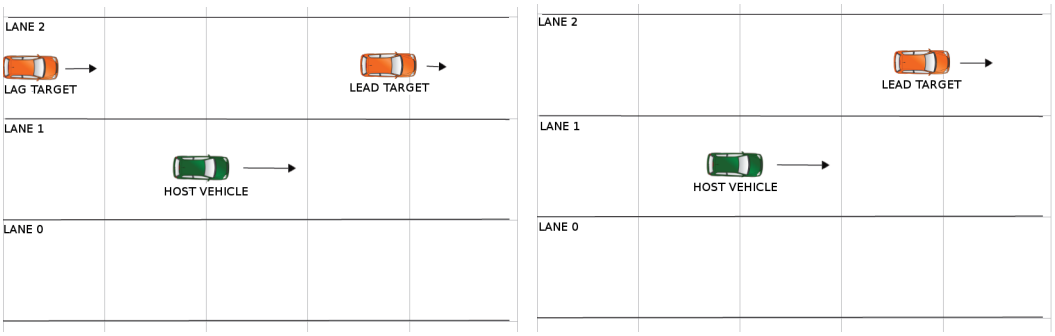
5.4.9 Simulation results

Simulations have been carried out to validate the trajectory-generation algorithms developed for lane-change maneuver. It computes the feasible range of time and acceleration parameters and generates the corresponding trajectories in both uniform velocity and accelerated modes. Following are some examples of simulated scenarios with results obtained from the proposed algorithm:

Algorithm 5 Accelerated lane-change.

Initialize $T_{min}, T_{max}, t_0 = T_1^{min}, a_{min}, a_{max}, a_{low}, a_{high}, l_{min} = 0$
for $t = t_0 : T_{max}$ in intervals of some constant k
    if (V_2 does not exist)
        if ($V_H \geq V_{lead}$), $t = T_{max}$
    $a_{min} = maximum\left(\frac{V_{lag}(1+t)-V_H t-d_{lag}}{0.5t^2}, \frac{V_{lead}t-V_H(1+t)+d_{lead}}{0.5t^2+t}\right)$
    $a_{max} = minimum\left(\frac{(V_{lag}-V_H)(2+t)-d_{lag}}{0.5t^2+2t}, \frac{(V_{lead}t-V_H)(2+t)+d_{lead}}{0.5t^2+2t}\right)$
    if (acceleration space exists within bound)
        lane-change Feasible with $a = [a_{min}, a_{max}]$ at time t
        if (acceleration space positive)
            exit the mode
        else
            compute length of path l
            if $l < l_m$, exit the mode
            else $l_m = l$ and continue
    else
        if (V_2 does not exist)
            lane-change not Feasible
            exit the mode

1. A scenario with two target vehicles and a host vehicle is considered in Fig. 5.12(a). By altering the velocities of the vehicles and the distance between each of them, different situations can be simulated. V_H, V_1 and V_2 are the velocities of the host, lead and lag target vehicles, respectively, and d_{lead} and d_{lag} are the lead and lag gaps, respectively. In the simulations, paths closer to the limits determined are marked in red/orange, safer ones in yellow and the recommended ones are marked in green.

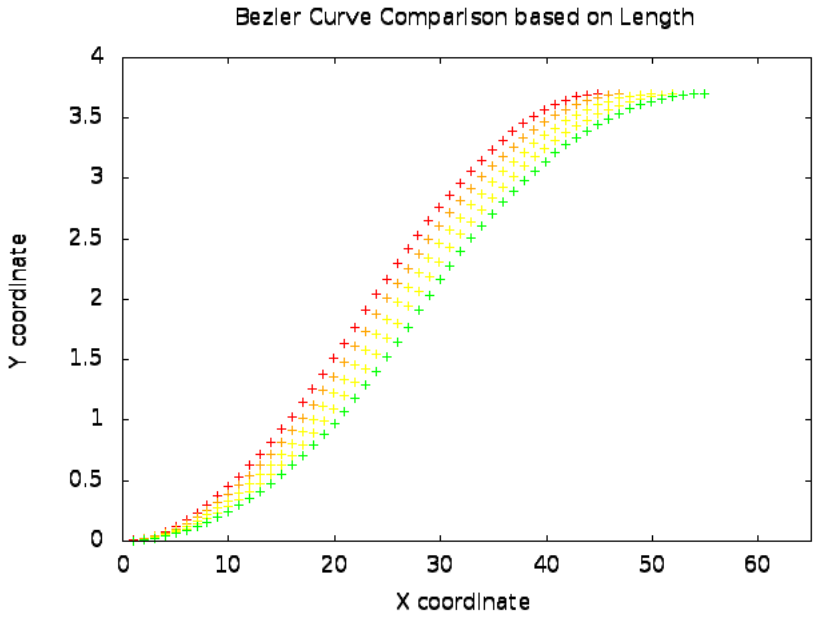


(a) Scenario 1. (b) Scenario 2.

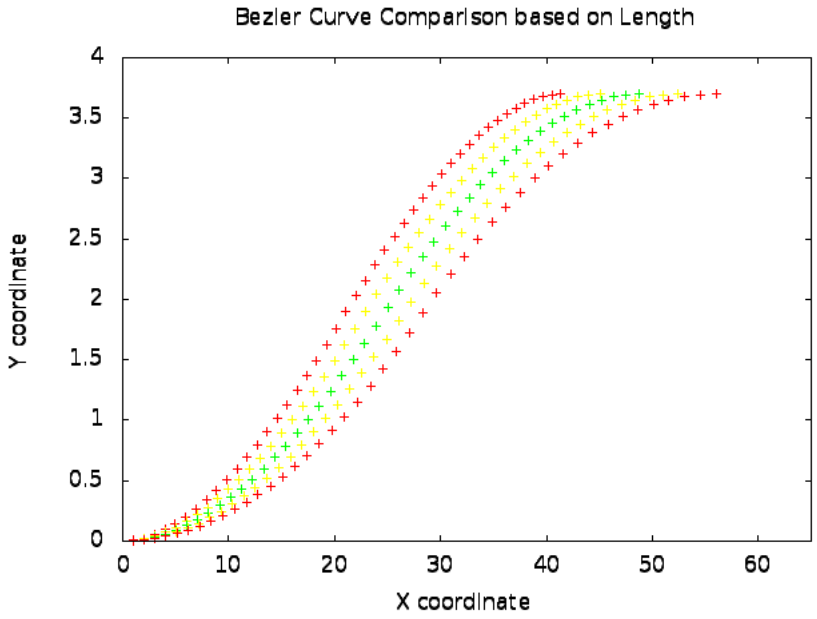
Fig. 5.12.: Simulation scenarios.

The simulation scenario in Fig. 5.12(a) is set up with $V_H = 10\ m/s, V_1 = 15\ m/s, V_2 = 13\ m/s, d_{lead} = 35\ m$, and $d_{lag} = 30\ m$. On solving the algorithm, we find the range of parameters as:

$$-0.34 \leq a \leq 1.5\ ;\quad T = 4.5\ s \quad \text{Accelerated mode}$$
$$V_H = 10\ m/s\ ; 4.5 \leq T \leq 5.6 \quad \text{Uniform velocity mode}$$
$$a = 0.5\ m/s^2\ ;\quad T = 4.5\ s$$
$$V_H = 10\ m/s\ ;\quad T = 5.6\ s$$

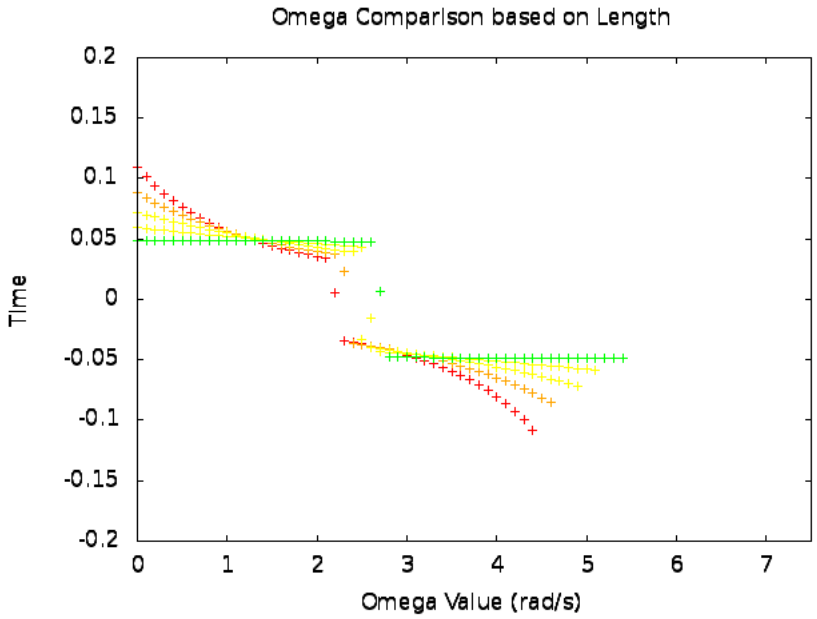


(a) Trajectories in uniform velocity mode.

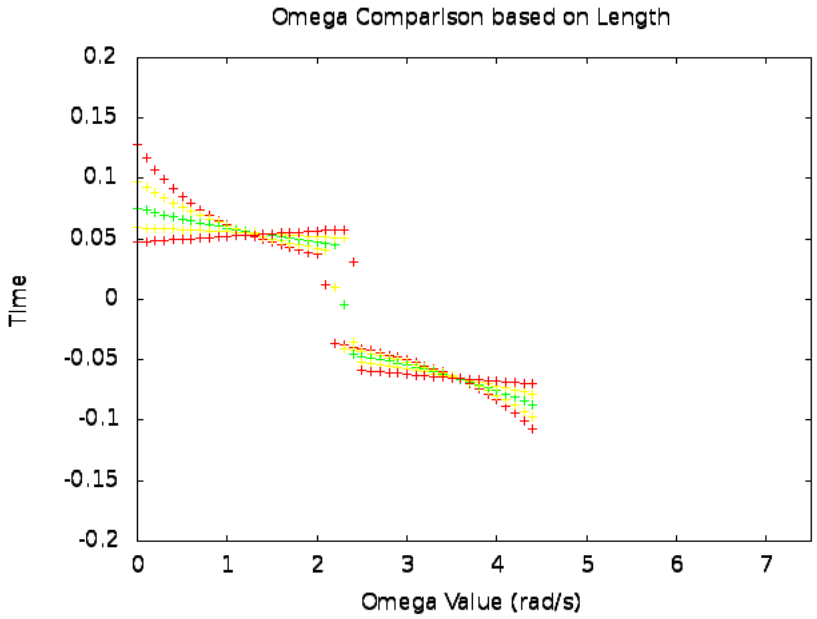


(b) Trajectories in accelerated mode.

Fig. 5.13.: Paths planned with possible acceleration and time values.



(a) Turn-rate variations in uniform velocity mode.



(b) Turn-rate variations in accelerated mode.

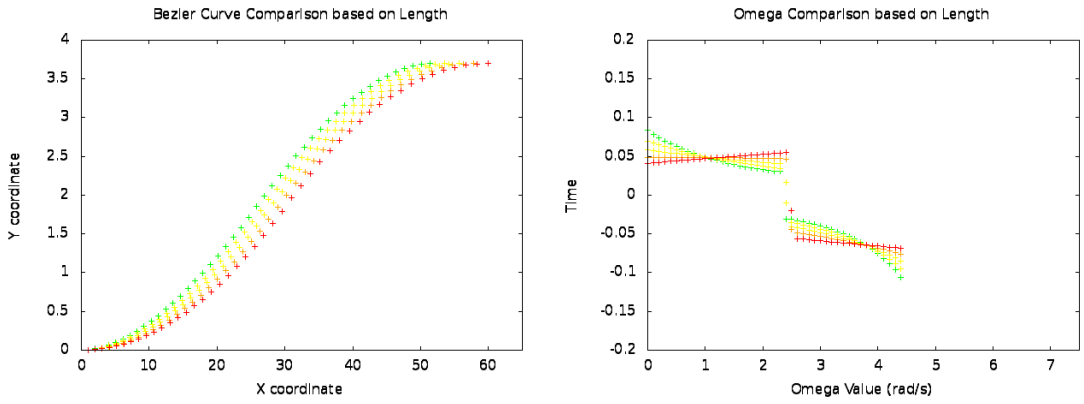
Fig. 5.14.: Turn-rate analysis.

Both accelerated and uniform-velocity modes of lane-change maneuvers are feasible in this scenario. Figures 5.13(a) and 5.13(b) show the generated trajectories in the uniform-velocity mode and the accelerated mode, respectively. The turn-rates obtained are within the limits (see Fig. 5.14). For the uniform-velocity mode, the solution with largest time is considered to be the optimum solution.

On decreasing the lag gap to 20 m in the same scenario (see Fig. 5.12(a)) and applying the lane-change algorithm, we obtain the range of parameters as:

$$0.65 \leq a \leq 1.5 \ ; \quad T = 4.5s \quad \text{Accelerated mode}$$

$$a = 0.65 m/s^2 \ ; \quad T = 4.5s$$



(a) Generation of trajectories.

(b) Turn-rate analysis.

Fig. 5.15.: Accelerated mode analysis for simulation scenario 2.

In this case, lane-change is feasible only in the accelerated mode. Figure 5.15 shows the trajectories generated and the corresponding turn-rate variations in the acceleration mode.

When the lag distance is further decreasd to $10\,m$, we observe that neither of the modes is feasible, and hence lane-change is not possible for the given values of velocities and gaps.

In a similar scenario, but with a new set of parameter values, $V_H = 7m/s, V_1 = 6m/s, V_2 = 8m/s, d_{lead} = 15m, d_{lag} = 30m$ are used for another simulation. The range of parameters obtained for this scenario are:

$$\begin{aligned} -1.23 \leq a \leq -0.1 \ ; \quad T = 7.5s \quad \text{Accelerated mode} \\ a = -0.1m/s^2 \ ; \quad T = 7.5s \end{aligned}$$

In this case, lane-change is possible only in the deceleration mode. The algorithm selects the lowest value of deceleration along with the largest value of time to obtain a safe, yet comfortable trajectory.

2. A slightly different scene is considered in Fig. 5.12(b) with no lag target vehicle and $V_H = 12m/s, V_1 = 15m/s, d_{lead} = 15m$.

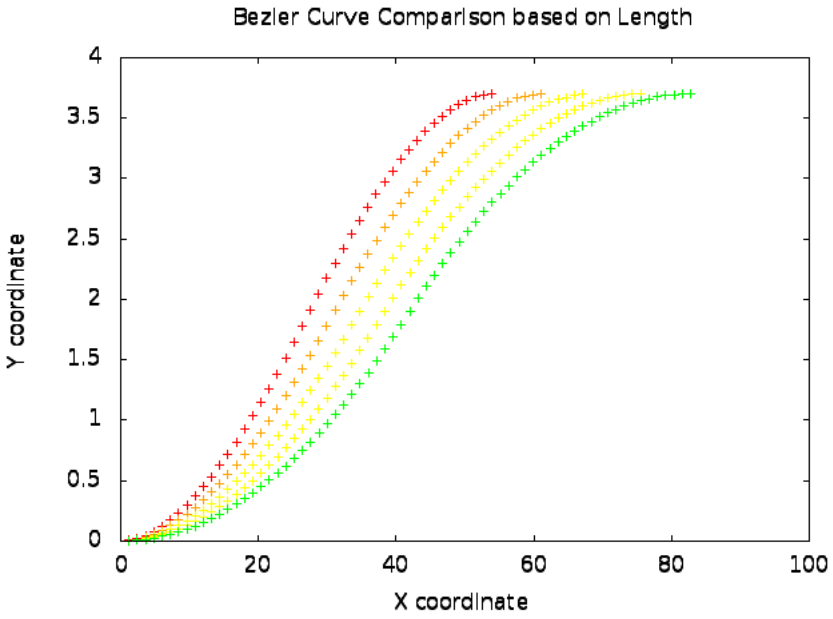
   On solving the algorithm, we find the range of parameters as:

$$\begin{aligned} -1.5 \leq a \leq 1.13 \ ; \quad T = 4.5s \quad \text{Accelerated mode} \\ V_H = 12\,m/s \ ; 4.5 \leq T \leq 7.5\,s \quad \text{Uniform velocity mode} \\ a = 0.5\,m/s^2 \ ; \quad T = 4.5\,s \\ V_H = 12\,m/s \ ; \quad T = 7.5\,s \end{aligned} \tag{5.61}$$

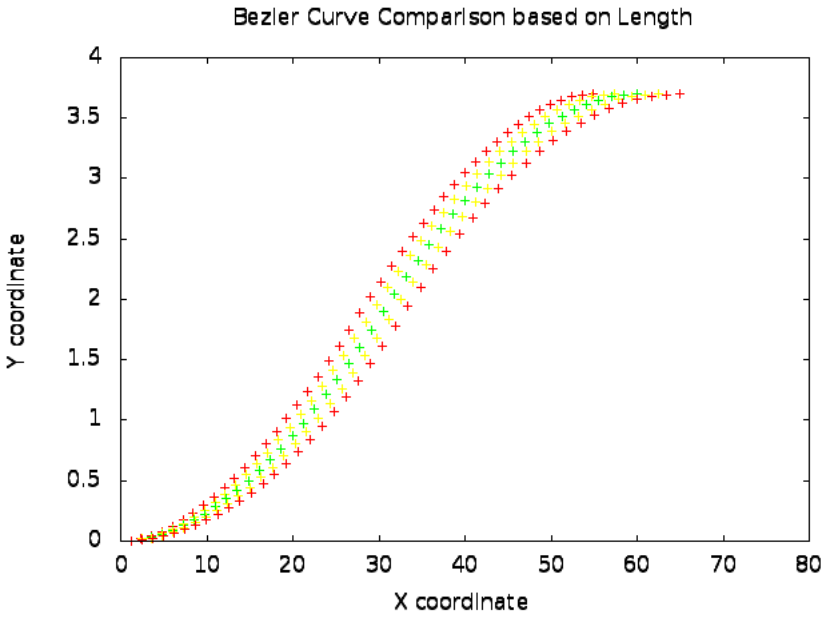
   In this case, the host vehicle can either perform a decelerated, accelerated or a uniform-velocity lane-change. Figure 5.16 shows the multiple paths that can be generated within the feasible set of acceleration values. Figure 5.17 illustrates the fact that turn-rates remain well within the limit during the maneuver.

Some common observations made from the simulation results are as follows:

1. During decelerated lane-change, the algorithm selects the lowest possible deceleration and a large time-value for performing the lane-change.

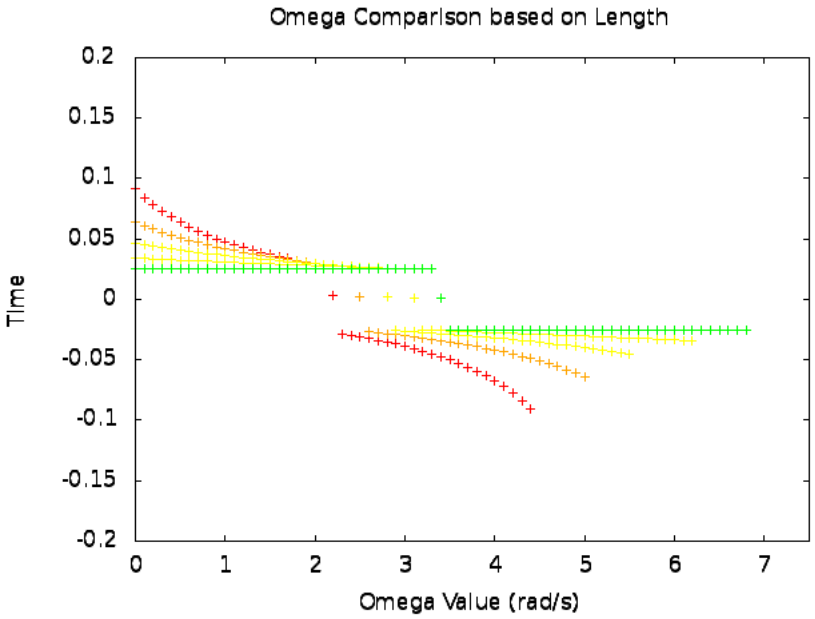


(a) Trajectories in uniform velocity mode.

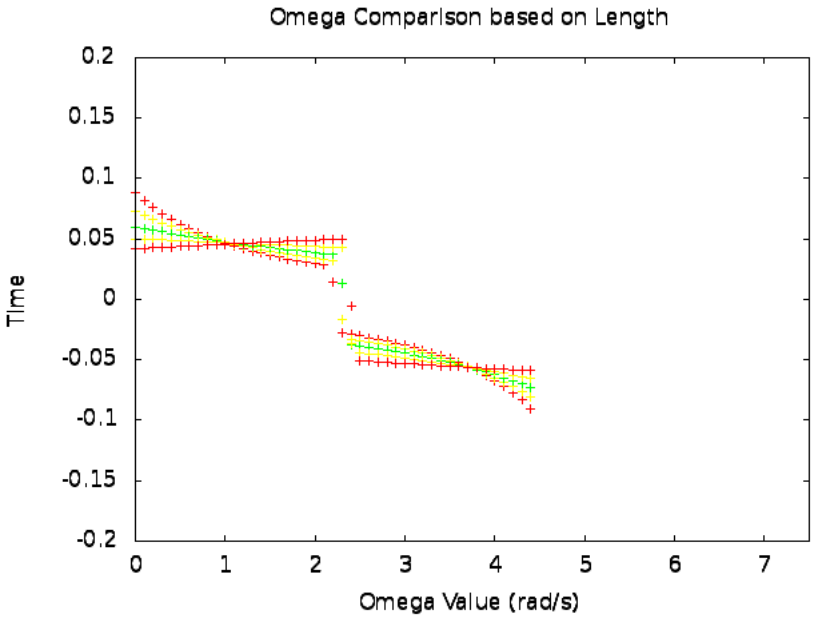


(b) Trajectories in accelerated mode.

Fig. 5.16.: Paths planned with possible acceleration and time values.



(a) Turn-rate variations in uniform velocity mode.



(b) Turn-rate variations in accelerated mode.

Fig. 5.17.: Turn-rate analysis.

2. If the entire acceleration range is feasible, then the algorithm tries to maintain an acceleration that would set the final velocity of host vehicle close to the lead target vehicle.

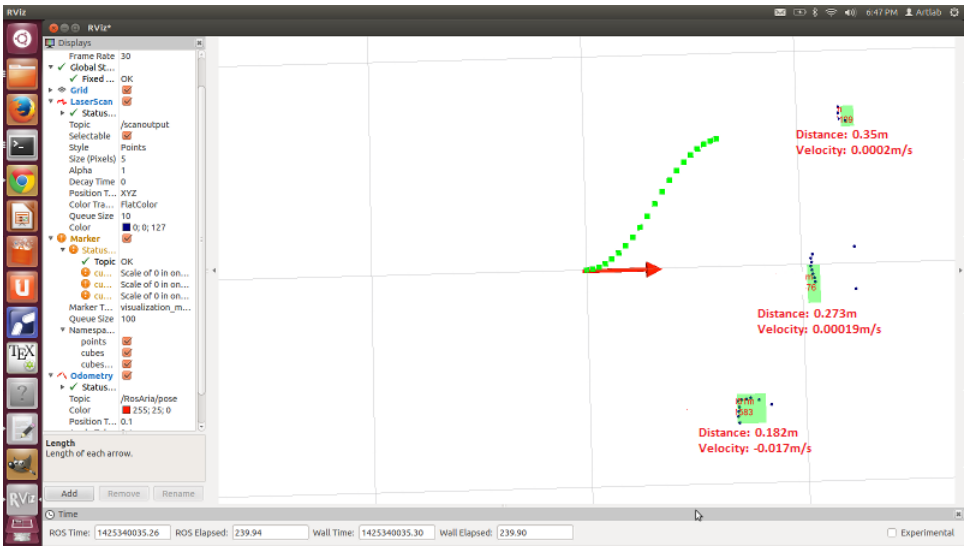
3. Sometimes the acceleration-range verifies the feasibility of the uniform velocity mode lane-change if the range of acceleration obtained contains zero-acceleration value. However, the converse is not true; that is, absence of this particular value of acceleration does not imply non-feasibility of uniform-velocity mode.

5.4.10 Experimentation with mobile robots

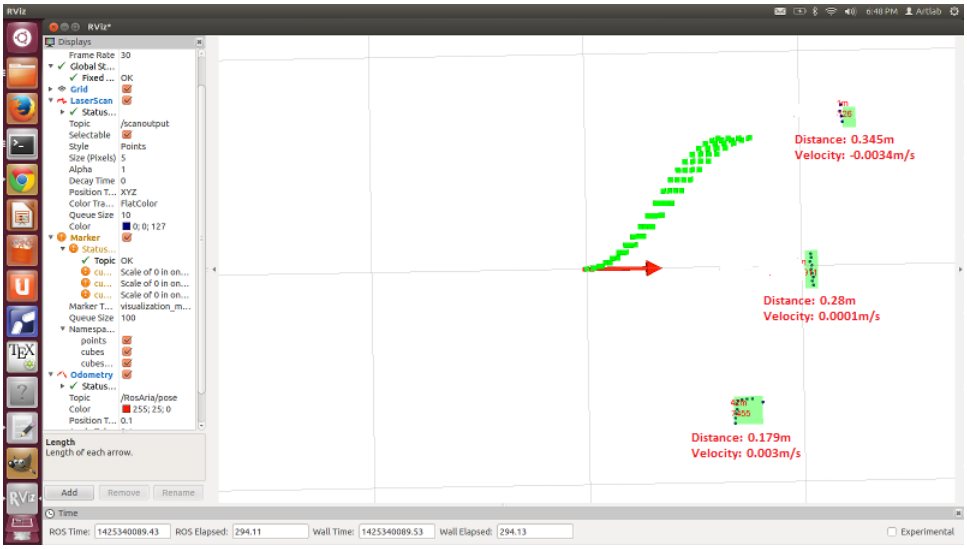
Based on the target-tracking estimates, lane-change paths can be generated when it is desired to perform a lane-change. As shown in Fig. 5.18, multiple paths (advisory mode) or a single path for a safe and convenient lane-change are generated. Once the desired trajectories are generated, controllers can be developed to ensure good lane-following technique for executing the lane-change maneuver.

An experiment was P3-DX performed using mobile robots as our autonomous vehicles, where the host vehicle is performed a lane-change with one lead vehicle in the same lane and another in the non-target lane. Figure 5.19 shows snapshots of a lane-changing maneuver using ROSViz visualization tool (see Fig. 5.19(a)), as well as images of the corresponding real-world scenario (see Fig. 5.19(b)).

Since the mobile robots are equipped with a 180° laser-scanner, they could not be used to validate the performance of the algorithm with lag vehicles. Therefore, scenarios with lead and lag vehicles are simulated using ROSViz simulation tool. Figures 5.20(a) and 5.20(b) show examples of trajectories generated in the advisory and co-pilot modes. Time T_0 and T_1 denote the start and the end time, respectively. The position of the target vehicles at given time instances, as well as their predicted position at the end of lane-change have been shown. Lane-changing trajectories are made to fit within the gap between the predicted positions of the lead and the lag vehicles.

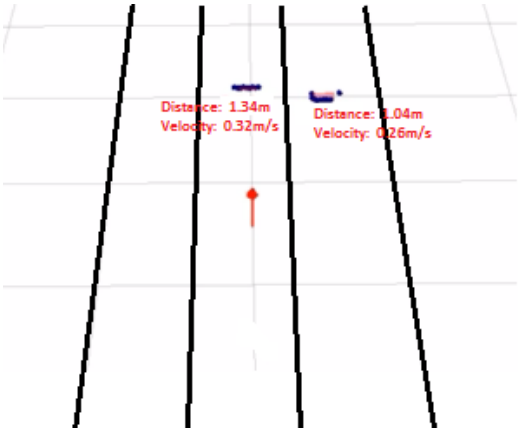


(a) Generated path in co-pilot mode.



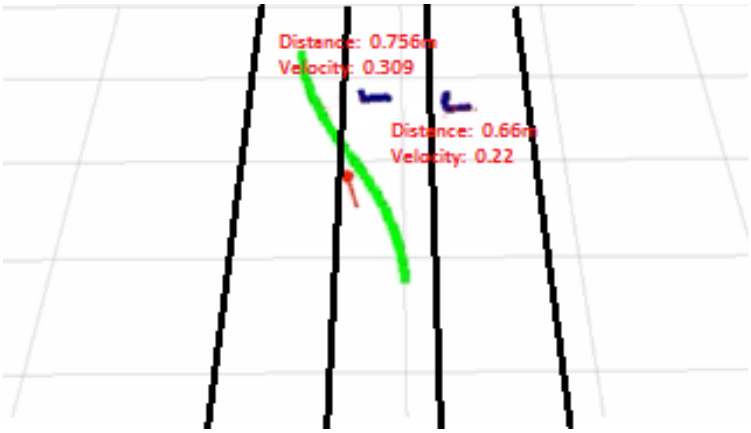
(b) Generated paths in advisory mode.

Fig. 5.18.: Generation of lane-changing trajectory.

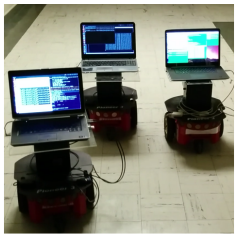


(a) Stage1: ROSViz Visualization.

(b) Stage1: Real-world scene.

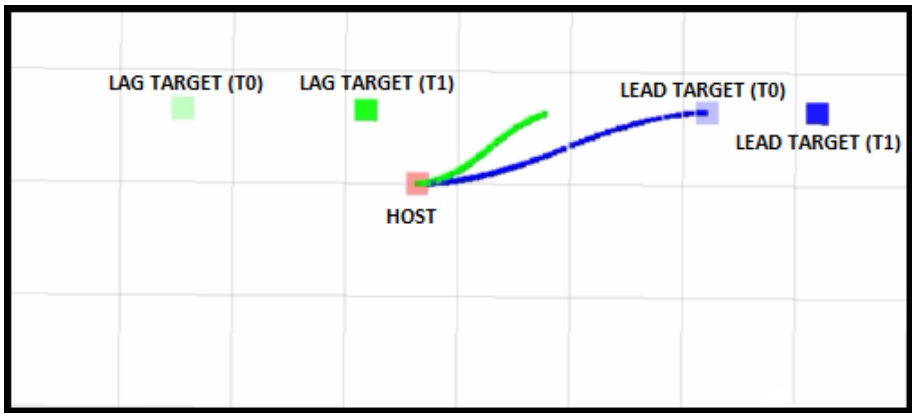


(c) Stage2: ROSViz Visualization.

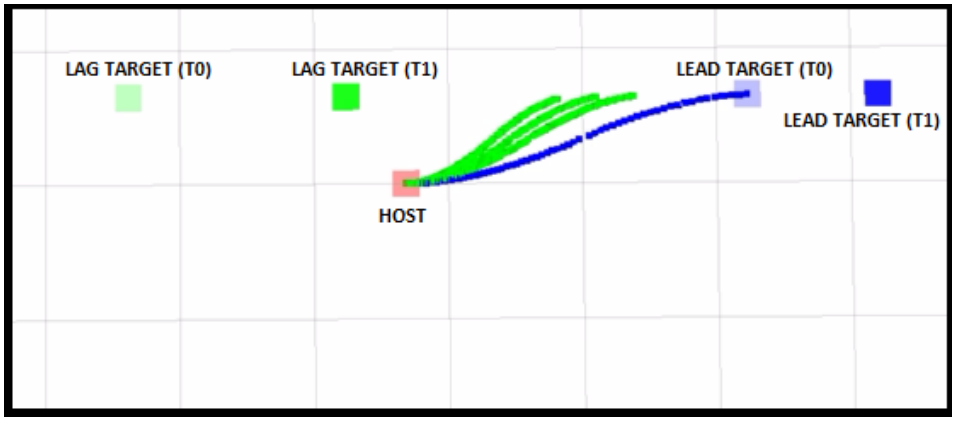


(d) Stage2: Real-world scene.

Fig. 5.19.: Lane-changing experiment using P3-DX in Assistive Robotics Laboratory.



(a) Generated path in co-pilot mode.



(b) Generated paths in advisory mode.

Fig. 5.20.: Generation of lane-changing trajectory.

5.5 Summary

In this chapter, lane-change trajectories have been planned based on the information obtained from the two-stage velocity estimator (discussed in Chapter 4). Piecewise Bezier curve has been used to generate lane-changing trajectories in both the uniform-velocity and the accelerated mode. The main objective of the algorithm is to ensure safe lane-change while maximizing the comfort-ride and minimizing the risk associated with the maneuver. Although no strict error metric has been used for testing the performance, evaluation of the turn-rate and acceleration values obtained from the path-planner has been carried out to validate the efficiency of the trajectory. Trajectory planning along with the proposed target tracking of neighboring vehicles form a complete lane-change trajectory generation algorithm to perform autonomous lane-change maneuver.

6. CONCLUSION AND FUTURE WORK

6.1 Introduction

It is important to perform multiple target-tracking of vehicles undergoing linear or curvilinear motion for an autonomous vehicle in a highway scenario. Lane-changing trajectory generation involves a decision-making process dependent on the behavior of the neighboring vehicles. The proposed 2-stage estimation algorithm tracks the target vehicles in the neighboring lanes during autonomous navigation (Chapter 4). Based on the velocity and lane information provided by the velocity estimator, a piecewise Bezier curve based method has been proposed for the generation of trajectory during lane-change (Chapter 5). A complete algorithm of lane-changing, including sensing and tracking of neighborhood vehicles, followed by planning and generation of lane-change trajectories, has been proposed in this research. Simulations and experimentation on a mobile robot platform have been carried out to validate the performance of the proposed algorithms.

6.2 Performance analysis of the algorithms

The algorithm developed for lane-changing trajectory generation caters to three different problems:

1. Vehicle detection: As described in Chapter 3, the first stage of a lane-change maneuver involves sensing of the environment, segmenting out the road section along with the associated components (lane markings, vehicles, obstacles) and performing feature extraction of the target vehicles. A modified Douglas-Pecker algorithm has been proposed to perform corner detection of the target vehicles. An accuracy up to 1.5cm has been achieved for the task of vehicle detection us-

ing a SICK LMS-200 laser-scanner. The integration of the proposed algorithm with existing camera-based segmentation and lane-detection algorithms has led to the classification of the detected vehicles into their respective lanes. Experimentation on a mobile robot platform with a 2D laser-scanner sensor has been carried out which validates the performance of the vehicle-detection algorithm.

2. Velocity estimation: A two-stage velocity estimator has been developed for the multiple target-tracking of maneuvering targets in the neighborhood of the host vehicle. An error analysis based on the outcomes of the proposed algorithm indicates that the root-mean-squared error of position is limited to about $5-6\,cm$ and velocity to about $0.25-0.3m/s$, with vehicles moving at a speed of $5-25m/s$ and performing lane-change within $4-7.5\,s$. For experimental purposes, mobile robots moving at speed of $0.2-0.5\,m/s$ have been considered. An average position error up to $0.5-0.75\,cm$ and velocity error upto $0.05-0.06\,m/s$ have been obtained. The algorithm locks turn-rate up to 10% accuracy within $3-4s$ for abrupt changes in turn-rate values. The multiple target-tracking aspect of the algorithm has been performed with very high NCA (Normalized correct association) values and low ICAR (Incorrect to Correct Association Ratio) values of about $0.25-0.35$. Computational time of the proposed algorithm has been restricted to about $15\,ms$. The algorithm performs well in tracking of the neighboring vehicles and aids in the trajectory generation for lane-change maneuvers.

3. Trajectory generation: Using the piecewise Bezier curve method, lane-changing trajectories have been generated in uniform speed and accelerated modes (Chapter 5). The velocity and lane information obtained from the velocity estimator is utilized in the generation of trajectories for the lane-change maneuver. The objective of the algorithm is to minimize the risk involved during the lane-change and generation of a trajectory that provides a good comfort-ride. Simulation

of various lane-changing scenarios have been carried out to validate the performance of the algorithm.

6.3 Future work

1. Road-curvature: In this research, minimal road curvature has been an important assumption. However, most of the highways have roads that are seldom straight, and hence an extension to the proposed algorithm would include road modeling with road curvature for better accuracy of the estimation algorithm.

2. Integration with camera/3D-LIDAR: This research deals with using laser-scanner for sensing the environment. However, one of the main challenges of using a 2D laser-scanner sensor is that the scanning is restricted to a 2D-plane, and hence sometimes important information might be missed out. This can be overcome by extension of the proposed algorithm by integrating camera-based algorithms for better precision. Another approach would include utilization of a 3D LIDAR for performing laser-scanning in three dimensions to capture a better model of the environment.

3. Lane-following algorithm: Complete autonomy can be achieved when the vehicle not only has the intelligence to generate a trajectory for the lane-change but also follow the trajectory to complete the maneuver. The proposed algorithm does not focus on the control of vehicle during the lane-changing maneuver. This implies that work can be done to develop a lane-change controller that would ensure the completion of the lane-change maneuver along the planned path.

4. Trajectory generation for the acceleration mode: In this research work, the complex problem of two-dimensional optimization for calculating an optimum time and acceleration has been reduced to a single variable optimization of acceleration for particular time. This provides scope of improvement in the

development of fast real-time optimization algorithms for real-time embedded processors that would provide an optimal solution in two dimensions for a better range of feasible paths.

5. Experimentation: In this research, experimentation has been carried out on a mobile robot platform. Although it validates the efficiency and performance of the algorithm to a certain extent, it is difficult to decide upon the computation complexity and latency using this simplified system. The platform can be upgraded to a real autonomous vehicle that would facilitate the real-time validation of the algorithm. The latest KITTI benchmark dataset [91] can also be used to extend the proposed algorithm to the case of using a 3D-LIDAR for scanning the environment.

The above discussion shows the direction in which this research can be extended. The proposed algorithm is generic, and would be able to adapt to most of the suggested changes for the advancement of research in autonomous navigation of vehicles.

LIST OF REFERENCES

LIST OF REFERENCES

[1] "Stanford's autonomous vehicle Stanley." http://es.autoblog.com/2005/10/11/vehculo-robot-de-la-universidad-de-stanford-gana-el-darpa-chall/. 2005.

[2] "Stanford's autonomous vehicle Junior." http://www.ros.org/news/2010/03/robots-using-ros-stanfords-junior.html. 2005.

[3] H. Jula, E. Kosmatopoulos, and P. Ioannou, "Collision avoidance analysis for lane changing and merging," *IEEE Transactions on Vehicular Technology*, vol. 49, pp. 2295–2308, Nov 2000.

[4] C. Hatipoglu, U. Ozguner, and K. Redmill, "Automated lane change controller design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, pp. 13–22, March 2003.

[5] "Lane-change Maneuvers." http://auto-era.net/general-car-topics/nissan-leaf-semi-autonomous-vehicle-hits-the-road-in-japan/. 2011.

[6] A. Balluchi, A. Bicchi, A. Balestrino, G. Casalinoz, A. B. A. Bicchiy, and A. B. G. Casalinoz, "Path tracking control for dubin's cars," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3123–3128, 1996.

[7] T. Maekawa, T. Noda, S. Tamura, T. Ozaki, and K. ichiro Machida, "Curvature continuous path generation for autonomous vehicle using b-spline curves," *Computer-Aided Design*, vol. 42, no. 4, pp. 350 – 359, 2010.

[8] C. Song-lin, X. Yi-bing, and Z. Ming, "K nearest neighbor joint possibility data association algorithm," in *Proc. International Conference on Information Engineering and Computer Science*, pp. 1–4, Dec 2010.

[9] J. wung Choi and G. H. Elkaim, "Bezier curve for trajectory guidance."

[10] J. Chen, P. Zhao, T. Mei, and H. Liang, "Lane change path planning based on piecewise bezier curve for autonomous vehicle," in *Proc. IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 17–22, July 2013.

[11] P. Petrov and F. Nashashibi, "Modeling and nonlinear adaptive control for autonomous vehicle overtaking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 1643–1656, Aug 2014.

[12] J. Naranjo, C. Gonzalez, R. Garcia, and T. de Pedro, "Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 438–450, Sept 2008.

[13] L. Guo, P.-S. Ge, M. Yue, and Y.-B. Zhao, "Lane changing trajectory planning and tracking controller design for intelligent vehicle running on curved road," in *Hindawi Publishing Corporation Mathematical Problems in Engineering*, 2014.

[14] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouveliere, "Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, pp. 589–606, Sept 2010.

[15] F. Nashashibi and A. Bargeton, "Laser-based vehicles tracking and classification using occlusion reasoning and confidence estimation," in *Proc. IEEE Intelligent Vehicles Symposium*, pp. 847–852, June 2008.

[16] C. Nemeth, P. Fearnhead, L. Mihaylova, and D. Vorley, "Particle learning methods for state and parameter estimation," in *IET Data Fusion Target Tracking Conference*, pp. 1–6, May 2012.

[17] D. Habermann and C. Garcia, "Obstacle detection and tracking using laser 2d," in *Latin American Robotics Symposium and Intelligent Robotic Meeting (LARS)*, pp. 120–125, Oct 2010.

[18] R. MacLachlan and C. Mertz, "Tracking of moving objects from a moving vehicle using a scanning laser rangefinder," in *IEEE Transactions on Intelligent Transportation Systems*, pp. 301–306, Sept 2006.

[19] "3D laser-scanning of the environment." `http://neilonpaper.com/tag/knight-rider/`. 2012.

[20] C. Ye and J. Borenstein, "Characterization of a 2d laser scanner for mobile robot obstacle negotiation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 2512–2518, 2002.

[21] S. Zeng, "A tracking system of multiple lidar sensors using scan point matching," *IEEE Transactions on Vehicular Technology*, vol. 62, pp. 2413–2420, July 2013.

[22] L. Gao, J. Xing, Z. Ma, J. Sha, and X. Meng, "Improved {IMM} algorithm for nonlinear maneuvering target tracking," *Procedia Engineering*, vol. 29, no. 0, pp. 4117 – 4123, 2012. 2012 International Workshop on Information and Electronics Engineering.

[23] V. Jilkov, "Design and comparison of mode-set adaptive imm algorithms for maneuvering target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, pp. 343–350, Jan 1999.

[24] Z. Zhong, H. Meng, and X. Wang, "Extended target tracking using an imm based rao-blackwellised unscented kalman filter," in *Proc. International Conference on Signal Processing*, pp. 2409–2412, Oct 2008.

[25] Y.-S. Kim and K.-S. Hong, "An imm algorithm for tracking maneuvering vehicles in an adaptive cruise control environment," *International Journal of Control, Automation, and Systems*, vol. 2, no. 3, pp. 310–318, 2004.

[26] X. Rong Li and Y. Bar-Shalom, "Design of an interacting multiple model algorithm for air traffic control tracking," *IEEE Transactions on Control Systems Technology*, vol. 1, pp. 186–194, Sep 1993.

[27] E. Semerdjiev, L. Mihaylova, and X. Li, "Variable- and fixed-structure augmented zmm algorithms using coordinated turn model," in *Proceedings of the Third International Conference on Information Fusion*, vol. 1, pp. MOD2/25–MOD2/32 vol.1, July 2000.

[28] P. Wu and X. Li, "Passive multi-sensor maneuvering target tracking based on ukf-imm algorithm," in *WASE International Conference on Information Engineering*, vol. 2, pp. 135–138, July 2009.

[29] B. Lee, J. Park, Y. Joo, and S. Jin, "Intelligent kalman filter for tracking a manoeuvring target," *IEE Proceedings on Radar, Sonar and Navigation*, vol. 151, pp. 344–350, Dec 2004.

[30] Wang-lingqun, Pan-shizhu, and Zheng-yingping, "Moving vehicle tracking based on unscented kalman filter algorithm," in *WRI World Congress on Computer Science and Information Engineering*, vol. 2, pp. 33–38, March 2009.

[31] X. Yuan, C. Han, Z. Duan, and M. Lei, "Adaptive turn rate estimation using range rate measurements," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, pp. 1532–1541, October 2006.

[32] F. Sun, Y. Ma, and X. E, "Maneuvering target tracking approaches based on turning rate estimation," in *International Symposium on Intelligent Information Technology Application*, vol. 2, pp. 213–216, Nov 2009.

[33] J. B. B. Gomes, "An Overview on Target Tracking Using Multiple Model Methods," Master's thesis, Instituto Superior Tecnico, Portugal, 2008.

[34] A. Gorji and M. Menhaj, "Multiple target tracking for mobile robots using the jpdaf algorithm," in *Proc. IEEE International Conference on Tools with Artificial Intelligence*, vol. 1, pp. 137–145, Oct 2007.

[35] "Adaptive Cruise Control." `http://en.wikipedia.org/wiki/Autonomous_cruise_control_system`. 2006.

[36] "Adaptive Cruise Control." `http://blog.oakridgeford.com/2014/03/28/fords-adaptive-cruise-control/`. 2012.

[37] "Collision Warning and Avoidance." `http://en.wikipedia.org/wiki/Collision_avoidance_system`. 2007.

[38] "Collision warning and mitigation." `http://www.motorauthority.com/news/1030038_ford-announces-new-radar-based-collision-avoidance-system`. 2012.

[39] "Automatic parking." `http://psipunk.com/valeo-park4u`. 2007.

[40] "Lane Departure Warning System." `http://en.wikipedia.org/wiki/Lane_departure_warning_system`. 2008.

[41] "Lane-keeping assist." `http://www.beachautomotive.com/blog/top-5-safety-checks/`. 2004.

[42] "Lane-changing assist." `http://inventorspot.com/articles/nhtsa_could_mandate_new_safety_features_2012_29915`. 2006.

[43] "Google self-driving car." http://inventorspot.com/articles/nhtsa_could_mandate_new_safety_features_2012_29915. 2006.

[44] "Google Self-driving Car." http://en.wikipedia.org/wiki/Google_driverless_car. 2010.

[45] Y. Zhang, C. Guo, H. Hu, S. Liu, and J. Chu, "An algorithm of the adaptive grid and fuzzy interacting multiple model," *Journal of Marine Science and Application*, vol. 13, no. 3, pp. 340–345, 2014.

[46] G. Hegeman, K. Brookhuis, and S. Hoogendoorn, "Opportunities of advanced driver assistance systems towards overtaking."

[47] M. Paldhe, "Software Architecture and Development for Controlling a HUBO Humanoid Robot," Master's thesis, Purdue University, West Lafayette, USA, 2014.

[48] "ROS node communication." http://barraq.github.io/fOSSa2012/slides.html. 2011.

[49] H. Wei, Z. Huang, Q. Yu, M. Liu, Y. Guan, and J. Tan, "Rgmp-ros: A real-time ros architecture of hybrid rtos and gpos on multi-core processor," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2482–2487, May 2014.

[50] "Pioneer3-DX monbile robot." http://www.mobilerobots.com/Accessories/PioneerGripper.aspx. 1997.

[51] "SICK-LMS200 laser-scanner sensor." http://www.pages.drexel.edu/~kws23/tutorials/sick/sick.html. 1990.

[52] "SICK-LMS200 demo program." http://homepages.engineering.auckland.ac.nz/~cyue001/tech/LMSintro.html. 1996.

[53] A. Mendes and U. Nunes, "Situation-based multi-target detection and tracking with laserscanner in outdoor semi-structured environment," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, pp. 88–93 vol.1, Sept 2004.

[54] G. Wang, Y. Zhou, G. Xu, X. Liu, and Y. Liu, "A novel lane changing algorithm with efficient method of lane detection," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2458–2463, Dec 2013.

[55] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen, "A novel lane detection based on geometrical model and gabor filter," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pp. 59–64, June 2010.

[56] P. Nunez, R. Vazquez-Martin, J. del Toro, A. Bandera, and F. Sandoval, "Feature extraction from laser scan data based on curvature estimation for mobile robotics," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1167–1172, May 2006.

[57] B. Fortin, R. Lherbier, and J.-C. Noyer, "Feature extraction in scanning laser range data using invariant parameters: Application to vehicle detection," *IEEE Transactions on Vehicular Technology*, vol. 61, pp. 3838–3850, Nov 2012.

[58] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1929–1934, Aug 2005.

[59] "Overview of RANSAC algorithm." `http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf`. 2000.

[60] "Linear rigression." `http://en.wikipedia.org/wiki/Linear_regression`. 2001.

[61] X. Li and V. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, pp. 1333–1364, Oct 2003.

[62] W. Gong, J. Li, and S. Wu, "An adaptive curvilinear model tracking algorithm based on the adaptive estimation of parameter," in *IET International Radar Conference*, pp. 1–4, April 2009.

[63] Y. shik Kim, K. shik Hong, Y. shik Kim, and K. shik Hong, "An imm algorithm for tracking maneuvering vehicles in an adaptive cruise control environment," *International Journal of Control, Automation, and Systems*, pp. 310–318, 2004.

[64] F. Sun, Y. Ma, and X. E, "Maneuvering target tracking approaches based on turning rate estimation," in *International Symposium on Intelligent Information Technology Application*, vol. 2, pp. 213–216, Nov 2009.

[65] M. Efe and D. Atherton, "Maneuvering target tracking using adaptive turn rate models in the interacting multiple model algorithm," in *Proceedings of the 35th IEEE Conference on Decision and Control*, vol. 3, pp. 3151–3156 vol.3, Dec 1996.

[66] T. Kirubarajan and Y. Bar-Shalom, "Kalman filter versus imm estimator: when do we need the latter?," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, pp. 1452–1457, Oct 2003.

[67] P. Konstantinova, A. Udvarev, and T. Semerdjiev, "A study of a target tracking algorithm using global nearest neighbor approach," in *Proceedings of the 4th International Conference on Computer Systems and Technologies*, CompSysTech '03, (New York, NY, USA), pp. 290–295, ACM, 2003.

[68] D. Schulz, W. Burgard, D. Fox, and A. Cremers, "Tracking multiple moving objects with a mobile robot," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–371–I–377 vol.1, 2001.

[69] Y. Bar-Shalom, X. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley, 2004.

[70] "Data Association Techniques." `http://www.micc.unifi.it/lisanti/downloads/data-association.pdf`. 2007.

[71] B. Habtemariam, R. Tharmarasa, T. Thayaparan, M. Mallick, and T. Kirubarajan, "A multiple-detection joint probabilistic data association filter," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, pp. 461–471, June 2013.

[72] Y. Bar-Shalom and L. A. U. E. University of California, *Multitarget-multisensor Tracking: Applications and Advances*. No. v. 3 in Artech House radar library, Artech House, 2000.

[73] F. Wang, M. Yang, and R. Yang, "Conflict-probability-estimation-based overtaking for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 366–370, June 2009.

[74] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, vol. 54, pp. 481–497, March 2009.

[75] "Basics of Kalman Filter." `http://en.wikipedia.org/wiki/Kalman_filter`. 2007.

[76] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[77] X. Li and Y. Bar-Shalom, "Mode-set adaptation in multiple-model estimators for hybrid systems," in *American Control Conference, 1992*, pp. 1794–1799, June 1992.

[78] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE Journal of Oceanic Engineering*, vol. 8, pp. 173–184, Jul 1983.

[79] W. Wu, P. Cao, and Z. Pan, "Maneuver target tracking based on kalman imm algorithm," in *Advances in Electrical Engineering and Automation* (A. Xie and X. Huang, eds.), vol. 139 of *Advances in Intelligent and Soft Computing*, pp. 479–483, Springer Berlin Heidelberg, 2012.

[80] "Interactive-Multiple-Model-based Estimator." `http://www.pce.hr/radar_systems/tracking/tracking.html`. 2006.

[81] C. Wonshik and T. Masayoshi, "Vehicle lane change maneuver in automated highway systems," in *California Partners for Advanced Transit and Highways (PATH)*, 1994.

[82] L. Wan, P. Raksincharoensak, K. Maeda, and M. Nagai, "Lane change behavior modeling for autonomous vehicles based on surroundings recognition," in *International journal of Automotive Engineerig (SAE Japan)*, 2011.

[83] N. S. North, "Intersection Algorithms based on Geometric Intervals," Master's thesis, Brigham Young University, Utah, USA, 2007.

[84] E. Nava-Yazdani and K. Polthier, "De casteljau's algorithm on manifolds," *Comput. Aided Geom. Des.*, vol. 30, pp. 722–732, Oct. 2013.

[85] "A Primer on Bezier Curves." `http://pomax.github.io/bezierinfo/`. 2011.

[86] L. Han, H. Yashiro, H. Nejad, Q. H. Do, and S. Mita, "Bezier curve based path planning for autonomous vehicle in urban environment," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pp. 1036–1042, June 2010.

[87] T. Toledo and D. Zohar, "Modeling duration of lane changes," in *Journal of the Transportation Research Board*, 2007.

[88] H. J. C, "Near-miss determination through use of a scale of danger," in *51st Annual Meeting of the Highway Research Board*, 1972.

[89] H. Park, C. S. Bhamidipati, and B. L. Smith, "Development and evaluation of an enhanced intelli-drivesm enabled lane changing advisory algorithm to address freeway merge conflict," in *Annual Meeting of the Transportation Research Board and in Review for Publication in the Transportation Research Record*, 2013.

[90] R. Schubert, K. Schulze, and G. Wanielik, "Situation assessment for automatic lane-change maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, pp. 607–616, Sept 2010.

[91] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.